PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-081401
(43)Date of publication of application : 28.03.1997

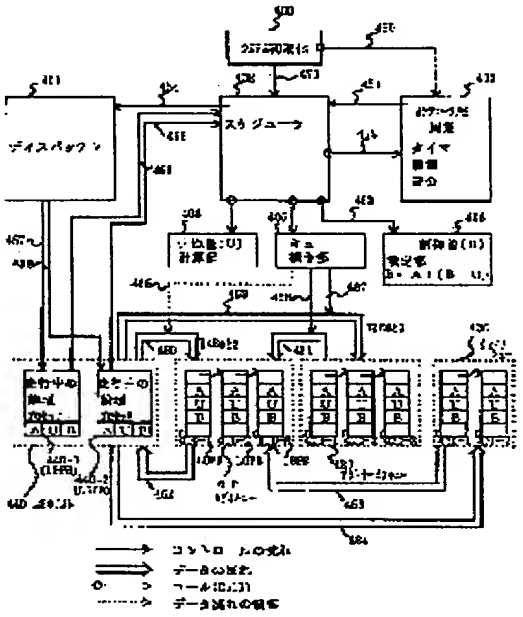
(51)Int.Cl. G06F 9/46
G06F 9/46

(21)Application number : 07-239516
(22)Date of filing : 19.09.1995

(71)Applicant : HITACHI LTD
(72)Inventor : UMENO HIDENORI
AMANO HIDEAKI
YAMAMOTO YASUO

(54) GLOBAL RESOURCE CAPPING METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To designate which of a global or local resource capping method is applied or not by limiting the CPU utilization factor of each virtual computer to a limit value or below in each small section. SOLUTION: A period timer interruption generation part 403 divides actual time into sections (about a second), and every time the section passes, a timer interruption is generated. A scheduler 402 updates a control value B by an expression B = initial control value A+(B-U). An interruption processing is performed for a running logical processing by the scheduler 402, service amount U is updated, further, a queuing is performed for a ready queue 410 or an out service queue 420 by the comparison of the service amount U after the update and the control value B. Subsequently, the processing of the setting part 406 of the control value B is performed. A control value B setting part 406 controls CPU service amount so as to be an initial control value A or below by fluctuating the control value B to be the upper limit value of the service amount of each logical processor for every section.



LEGAL STATUS

[Date of request for examination]	13.01.1999
[Date of sending the examiner's decision of rejection]	
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]	
[Date of final disposal for application]	
[Patent number]	3543441
[Date of registration]	16.04.2004
[Number of appeal against examiner's decision of rejection]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－81401

(43)公開日　平成９年(1997)３月28日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/46	３５０		Ｇ０６Ｆ　9/46	３５０	
	３４０			３４０Ｄ	

審査請求　未請求　請求項の数２　ＯＬ　（全 24 頁）

(21)出願番号　特願平7－239516

(22)出願日　平成７年(1995)９月19日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　梅野　英典
神奈川県秦野市掘山下１番地株式会社日立製作所汎用コンピュータ事業部内

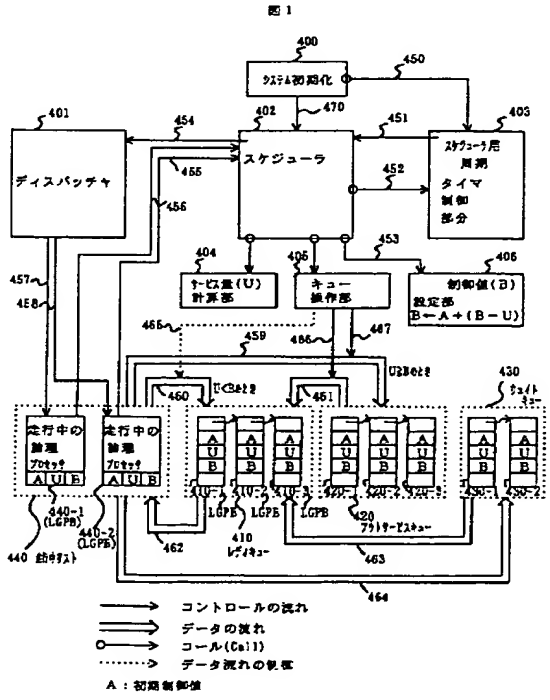
(72)発明者　天野　英昭
神奈川県秦野市掘山下１番地株式会社日立製作所汎用コンピュータ事業部内

(72)発明者　山本　保男
東京都千代田区内神田二丁目14番６号日立電子サービス　株式会社

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　大域的なリソースキャッピング方法

(57)【要約】
【目的】ＶＭのＣＰＵ利用率を、ユーザの指定量に抑える方式であるリソースキャッピングにおいて、短期（秒オーダ）の実時間ではなく、長期の実時間（分または時間のオーダ）おいて、指定量に抑える方式を提供する。
【構成】実時間において、小区間（例えば1秒）ごとにハイパバイザに割り込みを入れる手段、各小区間ごとに各ＶＭのＣＰＵ使用時間を測定する手段、各小区間において各ＶＭのＣＰＵ利用率を制限値以下に制限する手段からなる。



1

【 特許請求の範囲】

【 請求項1 】一台の実計算機システムの下で複数台のオペレーティングシステム(ＯＳ) を動かすことのできる仮想計算機システムにおいて、各ＯＳ のＣＰＵ利用率をある一定の値以下に制限するリソースキャッピング方法であって、各ＯＳ のＣＰＵサービス量を指定するステップと、それに基づいて各ＯＳ のＣＰＵサービス量の上限値である初期制御値を計算するステップと、実時間である観測時間を小さな時間である小区間に分割するステップと、ある上記のＯＳ のＣＰＵ利用率を、各該小区間において、ある制御値以下に抑える場合、該制御値を該ＯＳ の直前の小区間におけるＣＰＵ利用率によって動的に変動させるステップとにより、ある小区間においては、該ＯＳ のＣＰＵ利用率が初期制御値を越えることを許すが、該ＯＳ の全観測時間におけるＣＰＵ利用率を初期制御値以下に抑える大域的なリソースキャッピング方法。

【 請求項2 】一台の実計算機システムの下で複数台のオペレーティングシステム(ＯＳ) を動かすことのできる仮想計算機システムにおいて、各ＯＳ のＣＰＵ利用率をある一定の値以下に制限するリソースキャッピング方法であって、各ＯＳ のＣＰＵサービス量を指定するステップと、それに基づいて各ＯＳ のＣＰＵサービス量の上限値である初期制御値を計算するステップと、実時間である観測時間を小さな時間である小区間に分割するステップと、ある上記のＯＳ に対しては、そのＣＰＵ利用率を、各該小区間において、常に初期制御値以下に抑える局所的なリソースキャッピング方法を適用し、別のＯＳ に対しては、そのＣＰＵ利用率をある制御値以下に抑える場合、該制御値を該ＯＳ の直前の小区間におけるＣＰＵ利用率によって動的に変動させるステップとにより、ある小区間においては、該ＯＳ のＣＰＵ利用率が初期制御値を越えることを許すが、該ＯＳ の全観測時間におけるＣＰＵ利用率を初期制御値以下に抑える大域的なリソースキャッピング方法。

【 発明の詳細な説明】

【０００１】

【 産業上の利用分野】本発明は、一台の実計算機システムの下で複数台の仮想計算機(ＶＭ: Ｖirtual Machine) を同時に動かすことのできる仮想計算機システムのＣＰＵスケジュール方式に関する。各仮想計算機は、該実計算機と同等の機能を持ち、実計算機上で動くのと同じＯＳ を動作させることができる。この意味で、仮想計算機システムは一台の実計算機システムの下で複数台のＯＳ を同時に動かすことのできるシステムであるとも云える。仮想計算機システムを動かす実計算機システムをホスト 実計算機システムと呼ぶ。ＣＰＵスケジュール方式とは、同時に動作している各ＶＭに、いつ、どの程度のＣＰＵサービス量を与えるかということを決定する方式である。

【０００２】

2

【 従来の技術】仮想計算機システムの普及に伴い、その性能を向上させるために、いろいろなハードウェアによる支援機構が適用されてきた。仮想計算機システムは、ハイパバイザと呼ばれる制御プログラムを含む。ハイパバイザは、システムの主記憶、ＣＰＵ、Ｉ／Ｏ系というシステムリソースを各ＶＭに与え、各ＶＭを同時に動かすためにＣＰＵスケジュールを行う。すなわち、ハイパバイザは、そのＣＰＵスケジュールにおいて、各ＶＭの状態を管理し、それをいつ動かすか、それにどのくらいのＣＰＵサービス量を与えるかを決定する。

【０００３】仮想計算機システムを動かす実計算機システムは、マルチプロセッサシステムでも、ユニプロセッサシステムでも良い。ここでいう、マルチプロセッサシステムとは、主記憶装置を共有する２台以上のプロセッサからなり、ユニプロセッサシステムは1 台のプロセッサからなる。ここでいうプロセッサとは主記憶に格納されたプログラムを構成する機械命令をフェッチして実行していく機能を有する処理装置のことである。

【０００４】仮想計算機ＶＭの構成を図3 に示すが、詳しくは後述する。仮想計算機ＶＭは、それ自体ユニプロセッサモードまたは、マルチプロセッサモードであり得る。実計算機システムとの対応で云うと、ホスト 実計算機システムがＶＭに対応し、実プロセッサに対応するものを論理プロセッサと呼ぶ。すなわち、ユニプロセッサモードのＶＭは論理プロセッサ1 台からなり、マルチプロセッサモードのＶＭは、２台以上の論理プロセッサからなり、各論理プロセッサはＶＭにとっての主記憶領域を共有する。ＶＭの主記憶領域は、ホスト 実計算機システムの主記憶領域のある部分連続領域を、その主記憶領域として割り当てる場合もあるし、ハイパバイザが構成する仮想空間を、その主記憶領域として使用する場合もある。

【０００５】従来技術において、各ＶＭにサービス率を指定する機能がある。この機能は、以下のとおりである。たとえば、ＶＭ1、ＶＭ2、‥‥ＶＭN というN台(N は1 以上) の仮想計算機が同時に動作しているとしよう。その各々にサービス率Ｓ１、Ｓ２、‥‥、ＳN を指定したとする。各サービス率Ｓ iは、0以上である。このサービス率Ｓ１、Ｓ２、‥‥、ＳNは、各仮想計算機ＶＭ1、ＶＭ2、‥‥ＶＭNに割り当てるCPUサービス量を表す。さらに正確に云えば、各ＶＭは、待ち状態になることなく、ＣＰＵを与えられれば、いつでも、それを使い続けることができる状態(これをReady状態という) にあるとした場合、このサービス率指定は、各ＶＭ1、ＶＭ2、‥‥ＶＭNに与えられるＣＰＵサービス量の比がＳ１、Ｓ２、‥‥、ＳNとなるようにスケジュールすることを要求するものである。

【０００６】さらに従来技術においては、リソースキャッピング(resource capping) という機能がある。これについて以下に説明する。　サービス率制御において

3

は、各ＶＭが常にReady状態かまたは走行状態であるとき、そのサービス量の比が、指定サービス率の比になるように制御するものである。しかし、実際は、各ＶＭは、待ち状態になり得る。例えば、ＶＭ上のＯＳがＩ／Ｏ動作を要求し、その完了を待って待ち状態になる場合が、これに該当する。たとえば,今,ＶＭ1が、サービス率Ｓ1により規定されるサービス量Ａ１に達すること無く、待ち状態になったとしよう。そうすると、本来ＶＭ1に与えられるべきＣＰＵサービスが、特に他の指定も無い場合は、他のＶＭ2、・・・・ＶＭNに割当らることになる。そうすると、その　ＶＭ2、・・・・ＶＭNの中でＣＰＵを大量に使う負荷を持つもの（例えば、ＶＭ2とする）は、指定サービス率Ｓ2によって規定されるＣＰＵサービス量Ａ2を越えてサービスされることになる。

【０００７】たとえば、リソースキャッピングの例を、２台の仮想計算機ＶＭ1、ＶＭ2について考える。各サービス量をＳ1＝60、Ｓ2＝40とする。ホスト実計算機システムでの実プロセッサ台数を２台とし、その全処理能力を200としたとき、ＶＭ1に与えられる規定ＣＰＵサービス量Ａ1は、ＣＰＵ利用率にして２台の実プロセッサ合計で120%（すなわち1台の実プロセッサ当たり平均60%）、ＶＭ2に与えられる規定ＣＰＵサービス量Ａ2は、ＣＰＵ利用率にして２台の実プロセッサ合計で80%（すなわち1台の実プロセッサ当たり平均40%）、ということになる。　しかし、もし、ＶＭ2が待ち時間が多くて、ＣＰＵ利用率が、２台の実ＣＰＵ全体で 50% にしか達せず、ＶＭ1が常にReady状態、または、走行状態の負荷を持つとすると、本来ＶＭ2に与えられるべき 30% のＣＰＵ利用率が、ＶＭ1に与えられ、そのＣＰＵ利用率が 150% に達することとなる。これは、システム全体の性能向上としては、当然の処理方式であるが、ＶＭ1のユーザとしてはＣＰＵ利用率120%分の費用しか支払わないという契約を結んで使用したいというユーザもいる。このようなユーザに対しては、システムとして、ＶＭ1のＣＰＵ利用率を120%以下に制限する必要がある。この制限機能が、リソースキャッピングと呼ばれるものである。

【０００８】この従来のリソースキャッピングの問題点について以下に述べる。　あるＶＭのＣＰＵ利用率を規定値　Ａ%（Ａは0より大きな値とする）以下に抑えるということは、正確に考えると、二つの意味、すなわち、局所的に抑えると云う意味と、大域的に抑えると云う意味がある。この違いを図３９、図４０、図４１を用いて説明する。図３９は、あるＶＭのある負荷の本来のＣＰＵ利用率を実時間の経過を横軸にとって表したものである。９０がそのＣＰＵ利用率の推移を表す。値Ａ（９１）（%）が目的とする制御値であるとする。この負荷は時刻T0を境として、T0までは制御値Ａより、そのＣＰＵ利用率が低く、T0から移行は制御値Ａより高いとする。局所的に抑えるということは、観測時間（T1）

4

（例えば1時間）を各小区間（例えば1秒）に分割した場合、全ての小区間での　ＣＰＵ利用率を　Ａ% 以下に抑えるということである。したがて、この局所的リソースキャッピングを適用すると、この負荷のＣＰＵ利用率の経過は図４０に示すようになる。即ち時刻T0移行は制御値Ａで抑えられることになる。時刻T0より前では、元々、制御値Ａより低いので、そのまま低いままである。これを全体の観測時間T1で見るとこの負荷の全体のＣＰＵ利用率は、制御値Ａよりかなり低い値になってしまう。　一方、大域的に抑えるということは、局所的にＡ%を越えることがあっても良いが、長期の観測時間全体において　ＣＰＵ利用率を　Ａ% 以下に抑えるということである。この負荷に大域的リソースキャッピングを適用した場合のＣＰＵ利用率の経過を図４１に示す。ここでは、T0までが制御値Ａより低いので、その不足分をT0移降の制御値に加算することにより、T0以降では、制御値Ａを越えることを許しているが、全体の観測時間T1で見れば、全体のＣＰＵ利用率はＡで抑えられ、しかも、かなりＡに近い値にまで達することができる。このように局所的な意味で指定するか、大域的な意味で指定するかは、ユーザによるところである。しかし、従来のリソースキャッピングは、局所的な意味でしか指定できなかった。この局所的なリソースキャッピングについては、以下のような問題点がある。

【０００９】たとえば、ＣＰＵ利用率を　50% に抑えたい（Ａ＝50）とする。ある負荷を処理したとき、観測区間Ｔを、二つの同一長さの区間 T1, T2　に分割したとする。このとき、区間T1で、この負荷の本来の（リソースキャッピング制御を行わないときの）ＣＰＵ利用率が30%あるとすると、区間T1では、局所リソースキャッピングの適用、不適用にかかわらずＣＰＵ利用率は30%となる。次の区間T2で、たとえば該負荷のＣＰＵ利用率が本来 90% 有るものとすると、局所リソースキャッピングを適用すれば、ＣＰＵ利用率は50% に制限される。したがって、観測区間Ｔを通して考えると、ＣＰＵ利用率は40%に抑えらることになる。これは、ユーザから見れば抑え過ぎであるという意見もあるであろう。なぜなら、そのようなユーザにとっては、観測区間Ｔを通しての本来のＣＰＵ利用率は60% であり、それを、大域的に50%に制限するだけで良いという意見だからである。このようなユーザに対しては、大域的なリソースキャッピングを行う必要があるが、従来方式では、その手段がない。

【００１０】

【発明が解決しようとする課題】本発明の目的は、あるＶＭのＣＰＵ利用率を、ユーザの指定量に抑える方式であるリソースキャッピングにおいて、短期（秒のオーダ）の実時間ではなく、長期の実時間（分または時間のオーダ）において、該ＶＭのＣＰＵ利用率を該指定量に抑えるという方式を提供することである。各短期の区間

5

においては、該指定量を越えることが有っても良い。この方式を大域的なリソースキャッピングと呼ぶ。

【００１１】これに対し、従来のリソースキャッピングは、全ての短期区間において、該ＣＰＵ利用率を指定量以下に抑えるという局所的なリソースキャッピングである。

【００１２】本発明の別の目的は、各ＶＭ毎に、上記の大域的なリソースキャッピング方式を適用するか、局所的なリソースキャッピング方式を適用するかを指定可能とすることである。

【００１３】

【課題を解決するための手段】実時間において、小区間（例えば1秒）ごとに、ハイパバイザに割り込みを入れる手段、各小区間毎に、各ＶＭのＣＰＵ使用時間を測定する手段、各小区間において各ＶＭのＣＰＵ利用率の制限値を動的に指定する手段、各小区間において各ＶＭのＣＰＵ利用率を該制限値以下に制限する手段からなる。

【００１４】さらに、ユーザから、システム全体、または、各ＶＭごとにＣＰＵサービス率を指定する手段、システム全体、または、各ＶＭ毎に、局所的なリソースキャッピングを適用するのか、大域的なリソースキャッピングを適用するのかを指示する手段からなる。

【００１５】

【実施例】図1　は、本発明の基本的な構成図である。４００はシステム初期化を行う処理プログラムである。これは、ハイパバイザをたち上げるときに、システムで使用するデータを初期化する（詳細は後述する）。その初期化の一環として、周期タイマ割込み発生部４０３をコール（Call）する。４０３は、周期タイマと、それを制御するプログラムからなる。システム初期化４００は、線４５０が示す様に、周期タイマ割込み発生部４０３をコールし、全命令プロセッサに対して、周期的に、また、同期してタイマ割込みを発生するように設定する。そのタイマ割込みの周期Δｔは約1秒である。この周期はユーザが設定することもできる。この周期を区間と呼ぶ。４００はシステム初期化が終わった後、線４７０が示すように、スケジューラ４０２へ制御を渡す。スケジューラ４０２は、ハイパバイザの主要なプログラムである。スケジューラ４０２は、各仮想計算機ＶＭの各論理プロセッサをスケジュールする。すなわち、走行可能となった論理プロセッサをレディキュー（Ready Queue）４１０にキューイングすると共に、レディキュー４１０に登録された走行可能な論理プロセッサを選択して、線４５４が示すように、ディスパッチャ４０１に制御を渡す。ディスパッチャ４０１は、その選ばれた論理プロセッサ４４０－1または、４４０－2に、線４５７または、線４５８が示すように制御を渡し、走行させる。仮想計算機ＶＭは、図3に示す構成を持つが詳しくは後述する。仮想計算機ＶＭは、いくつかの（１台でも良い）論理プロセッサからなる。ハイパバイザは、論理

6

プロセッサの状態を管理するために、論理プロセッサタスク制御ブロックLGPB(Logical processor task block)を構成する（図7　に詳細を示し、後述する）。４４０は、走行中の論理プロセッサのリストであるが、４４０－1、４４０－2は、このLGPBであるが、走行中の論理プロセッサも表すものである。論理プロセッサが、走行可能ではあるが命令プロセッサがビジー（busy）であるが故に、それのサービスを待っている状態であるとき、これをレディー状態という。レディー状態の論理プロセッサに対応する制御ブロックLGPBが、レディキュー４１０にキューイングされる。４１０－1、４１０－2、４１０－3は、この制御ブロックLGPBを表す。走行中の論理プロセッサが、或事象の発生を待つことが必要になったとき、その論理プロセッサは、ウェイト（Wait）状態、すなわち、待ち状態に入る。待ち状態の論理プロセッサのLGPBは、ウェイトキュー（Wait Queue）４３０にキューイングされる。４３０－1、４３０－2がこの制御ブロックLGPBである。　各論理プロセッサタスク制御ブロックLGPBには、論理プロセッサのリソースデータ（レジスタや、PSWなど）の他に、命令プロセッサのサービスを受けた割合を表す値（Ｕ：これをＣＰＵサービス量と呼ぶ）と、リソースキャッピングの指定により、各論理プロセッサに対して決定された該サービス量の上限値ＡとＢが含まれる。これらを制御値Ａ，Ｂと呼ぶ。

【００１６】制御値Ａは、初期制御値であり、ユーザの指定値、アクティブな論理プロセッサの台数により、ＶＭのアクティベイト時に計算され、その後、新たに、ＶＭがアクティベイトまたはディアクティベイトされない限り一定である。一方制御量Ｂは、動的に変動する値である。４０６は、制御値Ｂ設定部であり、周期的に、この制御値Ｂを計算し直すプログラムである。

【００１７】この４０６の制御方法が、従来の方法と異なり、新方式を与え、本発明の主要部分を構成するものである。すなわち、各論理プロセッサのサービス量Ｕは、走行中の論理プロセッサが何らかの理由で中断されて、線４５５または線４５６が示すようにコントロールがスケジューラ４０２に移ったとき、スケジューラ４０２は、該中断された論理プロセッサのＣＰＵサービス量Ｕを計算し直す。このためにスケジューラ４０２は、サービス量計算部４０４をコールする。４０４は、中断されるまでに走行した命令プロセッサによるサービス時間を加算して、現在までのＣＰＵサービス量Ｕを該論理プロセッサについて求め、その制御ブロックLGPBに設定し直す。　さらに、ある論理プロセッサが中断されて、コントロールがスケジューラ４０２に移ったとき、そのサービス量Ｕを上記のように更新した後、スケジューラ４０２は、キュー操作部４０５をコールする。キュー操作部４０５は、該制御ブロックLGPB内の現在の該ＣＰＵサービス量Ｕを該制御値Ｂと比較する。キュー操作部４０５は、このときＵ＜Ｂであれば、いまだ制限値Ｂに達し

ていないので、該論理プロセッサタスク制御ブロックLGPBをレディキューにキューイングする。この制御ブロックLGPBの動きを表したものが線４６０であり、この動きを制御する働きが線４６５である。キュー操作部４０５は、そうでないとき、すなわちサービス量Uが制御値B以上となったとき、該論理プロセッサのサービス量は制御値Bに達したので、該論理プロセッサをアウトサービス状態とする。さらに、アウトサービス状態の論理プロセッサの制御ブロックLGPBは、ディスパッチの対象からはずすために、アウトサービスキュー４２０にキューイングする。このデータの動きが線４５９で表現される。さらに、このキューイングを制御する働きを線４６７で表す。制御ブロックLGPBである４２０－１、４２０－２、４２０－3は、このアウトサービスキュー４２０内の制御ブロックLGPBである。４０３は、周期タイマ割り込み発生部であり、実時間を区間（一秒程度）に分割して、その区間が経過する度にタイマ割り込みを発生させる。これを、スケジューラタイマ割り込みという。このスケジューラタイマ割り込みの発生により、コントロールが線４５１が示すようにスケジューラ４０２に移る。スケジューラ４０２は、制御値Bを以下の式により更新する。

【００１８】Ｂ＝Ａ＋（Ｂ－Ｕ）

この更新はレディーキュー４１０の全ての論理プロセッサタスク制御ブロックLGPBについて実行される。さらに、この更新は、アウトサービスキュー４２０の全ての論理プロセッサタスク制御ブロックLGPBについて実行される。さらに、この更新はウェイト・キュー４３０の全ての論理プロセッサタスク制御ブロックLGPBについても実行される。走行中の論理プロセッサは、この周期タイマ割り込み発生により、中断されて、スケジューラ４０２により、上記中断処理をされ、サービス量Uが更新され、さらに、更新後のサービス量Uと制御値Bとの比較により、レディーキュー４１０か、アウトサービスキュー４２０にキューイングされる。そのあと、上記の制御値Bの設定部４０６の処理が行われる。制御値AとBは、該各区間における制御値を表すものであり、したがって、上記のBの更新が終わった後、Uは0に設定される。この更新が終わった後、スケジューラ４０２は、キュー操作部４０５をコールし、アウトサービスキューの全ての論理プロセッサタスク制御ブロックLGPBをレディキュー４１０にキューイングし、それらの制御ブロックLGPBに対応する論理プロセッサを、次の該実時間の区間において、ディスパッチの対象とする。線４６１がこのデータの動きを表し、線４６６がこのキュー操作の制御の働きを表す。

【００１９】この発明の特徴は、制御値B設定部４０６の働きである。ここで、各論理プロセッサのサービス量の上限値である制御値Bを区間毎に変動させることにより、長期観測実時間において、ＣＰＵサービス量がＡ以下となるように制御することができる。たとえばある区間におけるＣＰＵサービス量が現在の制御値Ｂ０（制御値Bの初期値はAである）より、１０％少ない場合は、次の該実時間の区間における制御値Ｂ１は、Ａ＋10にする。この動的な制御値により、いくつかの区間においては、制御値Aを越えることがあっても、長期観測実時間における平均ＣＰＵサービス量をＡ以下とするという大域的なリソースキャッピングを実現することができる。また、制御値設定部４０６において制御値を常にAとすれば、該各区間全てにおいて、ＣＰＵサービス量がＡ以下となるので、従来の局所的リソースキャッピングを実現することもできる。さらに、ＶＭ毎に制御値設定部４０６の働きを変えることにより、ＶＭ毎に大域的リソースキャッピングと局所的リソースキャッピングとを別々に実現することもできる。

【００２０】図2は、ホスト実計算機システム１０を表す。仮想計算機ＶＭは、ホスト実計算機システム１０のイメージを持つ。仮想計算機ＶＭの構成図を図3に示す。ホスト実計算機システム１０は、一台以上の命令プロセッサを持っている。図2の場合は、4台の命令プロセッサ1－1、1－2、1－3、1－4を示している。命令プロセッサの台数は、これより多くても少なくても良い。各命令プロセッサは、線2－1、2－2、2－3、2－4を経由して、主記憶装置3につながる。仮想計算機システムは、一台以上の仮想計算機ＶＭからなるが、仮想計算機システム全体を制御するプログラムをハイパバイザと呼ぶ。ハイパバイザは、主記憶装置3のある領域9に常駐している。この領域はシステムの管理領域であり、各ＶＭはアクセスできない。ホスト実計算機システムは、外部記憶装置であるディスク装置8－1、8－2、・・・、8－Nを持つ。それらは、ディスク制御装置7に接続されており、その制御下にある。５は、Ｉ／Ｏプロセッサであり、主記憶装置3と線4を経由して接続され、さらに線6を経由してディスク制御装置7に接続される。Ｉ／Ｏプロセッサ5は、主記憶装置3と外部記憶装置（ディスク制御装置7がその例であるが、その他の外部記憶装置でも良い）との間のデータ転送を制御する。

【００２１】図3は、仮想計算機ＶＭ　２０の構成を示す。これは、ホスト実計算機システム１０のイメージを持つ。仮想計算機２０は、は一台以上の論理命令プロセッサ（論理プロセッサと略称する）を持っている。図2の場合は、２台の論理プロセッサ２１－1、２１－2を示している。論理プロセッサの台数は、これより多くても少なくても良い。各論理プロセッサは、線２２－1、２２－2を経由して、ＶＭの主記憶領域３０につながる。ＶＭの主記憶領域３０は、実主記憶装置3を分割した領域によって実現される。または、ハイパバイザが構成する仮想空間によって実現されることもある。論理プロセッサは、命令プロセッサのイメージを持ち、一台の

9

命令プロセッサにくくりつけられる、すなわち、それを専有することにより実現するか、または、他の論理プロセッサとの間で一台の命令プロセッサを実時間を分割して共有するかして実現される。仮想計算機システムは、一台以上の仮想計算機ＶＭからなるが、仮想計算機システム全体を制御するプログラムをハイパバイザと呼ぶ。ハイパバイザは、主記憶装置3 のある領域9 に常駐している。この領域はシステムの管理領域であり、各ＶＭはアクセスできない。仮想計算機ＶＭ　２０は、外部記憶装置である論理ディスク装置８０－１、８０－２、・・・、８０－Ｎを持つ。これらは、実のディスク装置8 －１、８－2、・・・、８－Ｎを専有するか、または、論理プロセッサ間で共有するかによって実現される。それらは、論理ディスク制御装置７０に接続されており、その制御下にある。論理ディスク制御装置７０は、実ディスク制御装置7 を専有するか、または、論理プロセッサ間で共有するかによって実現される。５０は、論理I ／Ｏプロセッサであり、ＶＭの主記憶領域３０と線４０を経由して接続され、さらに線６０を経由して論理ディスク制御装置７０に接続される。論理I ／Ｏプロセッサ５０、実I ／Ｏプロセッサ5 を専有または共有することにより実現される。論理I ／Ｏプロセッサ５０は、主記憶領域３０と外部記憶装置（論理ディスク制御装置７０がその例であるが、その他の外部記憶装置でも良い）との間のデータ転送を制御する。以上の仮想計算機ＶＭの実現方式は、従来と同じ方式である。

【００２２】図4 はＶＭの定義画面　１００を表す。１０１は、ＶＭの名称を定義するフィールドであり、ＶＭ１、ＶＭ2 、ＶＭ3 、ＶＭ4　の4 台のＶＭが定義されている。定義台数はこれより多くても少なくても良い。１０２は各ＶＭのサービス率を定義する部分であり、ＶＭ1 、ＶＭ2 、ＶＭ3 、ＶＭ4 の各サービス率が、それぞれ、Ｓ１、Ｓ２、Ｓ３、Ｓ４であることを示している。このサービス率は、ユーザが指定することができる。１０３は、各ＶＭの論理プロセッサＬＧＰ（ p,q）を定義するフィールドである。１０３の場合、ＶＭ1 は　論理プロセッサＬＧＰ（１、１）、ＬＧＰ（１、２）、ＬＧＰ（１、３）、ＬＧＰ（１、４）の4 台の論理プロセッサからなる。ＶＭ2 は　論理プロセッサ　ＬＧＰ（２、１）、ＬＧＰ（２、２）、ＬＧＰ（２、３）　の3 台の論理プロセッサからなる。ＶＭ3 は　論理プロセッサ　ＬＧＰ（３、１）、ＬＧＰ（３、２）の2 台の論理プロセッサからなる。ＶＭ4 は　論理プロセッサ　ＬＧＰ（４、１）、ＬＧＰ（４、２）の2 台の論理プロセッサからなる。各ＶＭの論理プロセッサの台数はこれより多くても少なくても良い。このＶＭの定義画面は従来と同様の方法であるが、１０８が従来と異なるフィールドであり、各ＶＭ毎に大域的なリソースキャッピングを指定する（ RC=YG）か、局所的なリソースキャッピングを指定する（ RC=YL）か、リソースキャッピングを

10

指定しない（ブランク）かを示す。

【００２３】図5 について説明する。１０７は、実時間の分割の方法について表したものである。１０４は、実時間τの流れを表す。１０６は短い時間Δｔ（ 1秒程度、マイクロ秒 i.e. μsec 単位）である。Δｔの大きさはシステムパラメタとして変更可能である。実時間τは、この短い時間　Δｔ　１０６　（これを以後区間と呼ぶ）によって分割して考えられる。０の時点がスタート時点を表し、システムの立ち上げ時に、全命令プロセッサ共通に、スタート 時点0 が設定される。１０５－1 は第1 区間である。１０５－2 は第2 区間である。１０５－i は、　第ｉ区間である。後述するが、この実時間の分割は全命令プロセッサに共通であり、システム立ち上げ時に、各区間での時間経過毎に、全命令プロセッサで同期してタイマの割込みが発生するように設定される。

【００２４】図6 は使用する記号の説明である。即ちＣ（ p,q）は、論理プロセッサＬＧＰ（ p,q）の上記区間ΔｔにおけるＣＰＵビジィータイム（ busy time）で単位は、μsecである。この値は、当然、区間によって異なるが、同じ記号で表す。Δｔは上記の区間の時間的長さであり、単位は、μsecである。　Ａ１（ %）はＶＭ1 のＣＰＵ利用率の初期制御値である。Ａ２（ %）はＶＭ2 のＣＰＵ利用率の初期制御値である。ＡＮ（ %）はＶＭＮのＣＰＵ利用率の初期制御値である。Ａ（ p,q）（ %）は、論理プロセッサ　ＬＧＰ（ p,q）のＣＰＵ利用率の初期制御値である。このＡ１、Ａ２、・・・、ＡＮ、Ａ（ p,q）,p=1,2,･･･; q=1,2,･･･、とＳ１、Ｓ２、・・・、ＳＮ、との関係は以下の通りである。

【００２５】ホスト 実計算機システム１０における命令プロセッサの台数をＫ台（ Ｋは1 以上）とすると、このＫ台の命令プロセッサの台数分の全処理能力（ Ｐ）を、Ｓ１、Ｓ２、・・・、ＳＮで比例配分した値をＡ１、Ａ２、・・・、ＡＮとする。このＡ１、Ａ２、・・・、ＡＮをそれぞれＶＭ1 、ＶＭ2 、・・・、ＶＭＮの制御値と呼ぶ。この全処理能力（ Ｐ）は、１００で表しても良いし、１００＊Ｋで表しても良い。ＶＭを構成する各論理プロセッサには均等割りした処理能力が与えられる。今、分かり易くするために、1 台の実命令プロセッサの処理能力を１００として考える。

【００２６】各ＶＭｐの　制御値　Ａｐに対して、p=1,2,3･･･

Ａ（ p,q）＝Ａｐ／（ ＶＭｐの論理プロセッサの台数）

ｑ＝１，２，３，・・・とする。これを、論理プロセッサ　ＬＧＰ（ｐ，ｑ）　の制御値と呼ぶ。

【００２７】ただし、Ａ（ p,q）が100を越える場合は、以下の補正に従う。その越えた値をD(p,q)とする。Ａ（ p,q）＝１００として、ＶＭp 全体での越えた値Dpは、

Dp=D(p,1)+D(p,2)+D(p,3)+･･･

11

となる。この Dp の値を、ＶMp以外の{ＶMr}のサービス率{Ｓr}で比例配分する。このとき、論理プロセッサの制御値が既に100に達している場合は、そのＶMは比例配分対象から外す。この比例配分した値を、各{ ＶMr} の制御値{ Ａr} に加える。こうして更新された制御値ＡrをＶMrの論理プロセッサ台数で割った値を論理プロセッサＬＧＰ(r,q) の制御値とする(q=1,2,3･･･) 。

【００２８】この補正を100を越える各Ａ(p,q) について繰り返し、対象のＶMが無くなったら終わりとする。以上の制御値の計算方法は、従来と類似の方法であるが、補正の方法は独自の方式である。

【００２９】図7 を説明する。１１０は、論理プロセッサタスク制御ブロック LGPBを表す。

【００３０】ハイパバイザは、論理プロセッサの状態を管理したり、ディスパッチしたり、走行させたりするのに、その主記憶中に論理プロセッサ制御ブロック LGPBを構成する。論理プロセッサ1 台に対して一つの制御ブロック LGPBを構成する。これは、ＶMをアクティベイトしたときに、その各論理プロセッサに対して構成される。

【００３１】制御ブロック LGPBは、状態フラグ１１１、論理プロセッサ番号１１２、論理プロセッサリソースデータ１１３、制御データ１１４を含む。状態フラグ１１１は、論理プロセッサの状態 S を含む。１１５にこの記号の意味を述べる。S=0のとき論理プロセッサはスリープ(sleep) 状態である。S=1のとき論理プロセッサはレディ(ready) 状態である。S=2 のとき論理プロセッサはアウトサービス(out-service) 状態である。S=3のとき論理プロセッサはウエィト(Wait) 状態である。スリープ状態とは、まだ論理プロセッサがスタートされていない状態である。レディ状態とは、走行可能であるが命令プロセッサが使用中であるために、その割当を待っている状態である。アウトサービス状態とは、論理プロセッサがその制御値に達するＣＰＵサービス量を受けたため、一時的に、ディスパッチ対象から外された状態である。ウェイト状態とは、論理プロセッサが或事象の発声を待っている状態である。この中でディスパッチ対象となるのは、レディ状態の論理プロセッサだけである。１１２は、ＶM番号p (=1,2,3,･･･) とそのＶMにおける論理プロセッサ番号q (=1,2,3,･･･) を含む。論理プロセッサリソースデータ１１３は、ゲスト汎用レジスタデータ１１３－1 、ゲスト制御レジスタデータ１１３－2 、ゲストＰＳＷデータ１１３－3 、ゲストＣＰＵタイマデータ１１３－4 を含む。ゲストＰＳＷデータ１１３－3 は、システムマスクＳM(ビット０からビット７まで) や、次に実行する命令のアドレスＮＩＡ(ビット３２から６３まで) を含む。システムマスクＳMの各ビットは、割り込みの可能性を制御するビットを含む。

【００３２】図8 は、論理プロセッサタスク制御ブロック LGPB の中に含まれる制御データ１１４の中のデータ

12

を示す。１１４－1 は、論理プロセッサ (p,q) のタイムスライス初期値 TS0(p,q)(単位はμsec) である。ビット０は符号ビットであり、1のときは負の値を表し、ビット５１が、1μsec の位である。タイムスライスの標準値は、システム立ち上げ時に決定されるが、その件は後述する。１１４－2 は、論理プロセッサ (p,q) の残りのタイムスライス TS(p,q)(単位はμsec) である。ビット０は符号ビットであり、1のときは負の値を表し、ビット５１が、1μsec の位である。１１４－3 は、論理プロセッサ (p,q) が実時間の区間において受けたＣＰＵサービス量を表すデータ C(p,q)(単位はμsec) を含む。ビット０は符号ビットであり、1のときは負の値を表し、ビット５１が、1μsec の位である。これらのデータは、ＣＰＵタイマと同じフォーマットである。１１４－4 は、論理プロセッサ (p,q)の初期制御値Ａ (p,q)(%) を含む。このＡ (p,q)は、前述の如く図4の１０２に示す各ＶMのサービス率Ｓ１、Ｓ２、･･･、ＳＮ によって決定される論理プロセッサのサービス量の上限値である。１１４－5 は、論理プロセッサ (p,q)の現在の制御値Ｂ (p,q)(%) を含む。Ｂ (p,q)の初期値はＡ (p,q)である。１１４－6 は、現在の区間における論理プロセッサ (p,q)のＣＰＵ利用率(%) U(p,q)を含む。区間 i のB(p,q)は、Ａ(p,q) に、直前の区間の 差分 Ｂ (p,q)－Ｕ (p,q) を加えた値によって、求められる。１１４－7 は現在の区間ＣＰＵ充足度D(p,q) (%) を表す。D(p,q)は、現在の区間における制御値Ｂ (p,q)に対する論理プロセッサ (p,q)のＣＰＵ利用率の割合、すなわち、

D(p,q)=(U(p,q)/B(p,q))*100 (%)

である。１１４－8 は統計データ不当インディケータ(1ビットの値sinv(p,q)) であり、sinv(p,q)=1のときは、上記の１１４－2 (TS(p,q)) 、１１４－3 (C(p,q)) 、１１４－6(U(p,q)) 、１１４－7(D(p,q))が古いデータである、すなわち、最新のデータではないことを表す。初期値は0 である。１１４－9 はリソースキャッピング制御データ rcであり、この論理プロセッサの属するＶMがリソースキャッピングの指定無し(rc=0) か、局所的なリソースキャッピングを指定している(rc=1) か、大域的なリソースキャッピングを指定している(rc=2) かを表す。

【００３３】図9 は、現在走行中の論理プロセッサの制御データを表す。図１０は、走行中の論理プロセッサ制御ブロック(LGPB) のアドレスレジスタ３０７、及び、その制御ブロック１１０－R (LGPB-R) を表す。１１０－R は、該当のＶM番号p及び論理プロセッサqを表す。１１４－R は、その制御データであり、図9 に示す制御データを含む。すなわち、１１４－1 －R は、走行中の論理プロセッサ (p,q) のタイムスライス初期値 TS0__R (p,q)(単位はμsec) である。ビット０は符号ビットであり、1のときは負の値を表し、ビット５１が、1μsec

13

の位である。１１４－2 －Rは、走行中の論理プロセッサ(p,q) の残りのタイムスライスTS_R(p,q)(単位はμsec) である。ビット０は符号ビットであり、1のときは負の値を表し、ビット５１が、1μsec の位である。１１４－3 －Rは、走行中の論理プロセッサ (p,q) が実時間の区間において受けたＣＰＵサービス量を表すデータC_R(p,q)(単位はμsec) を含む。ビット０は符号ビットであり、1のときは負の値を表し、ビット５１が、1μsec の位である。これらのデータは、ＣＰＵタイマと同じフォーマットである。 １１４－4 －Rは、走行中の論理プロセッサ (p,q)の初期制御値 A_R(p,q)(%) を含む。このA_R(p,q)は、前述の如く図4の１０２に示す各ＶＭのサービス率Ｓ１、Ｓ２、・・・、ＳＮ によって決定される論理プロセッサのサービス量の上限値である。１１４－5 －Rは、走行中の論理プロセッサ (p,q)の現在の制御値B_R(p,q)(%) を含む。B_R(p,q)の初期値はA_R(p,q)である。１１４－6 -Rは、現在の区間における走行中の論理プロセッサ (p,q)のＣＰＵ利用率U_R(p,q) (%)を含む。区間 i のB_R(p,q)は、A_R(p,q) に、直前の区間の 差分 B_R(p,q)－U_R(p,q) を加えた値によって、求められる。１１４－7 －Rは、走行中の論理プロセッサの現在の 区間ＣＰＵ充足度D_R(p,q) (%) を表す。D_R(p,q)は、現在の区間における制御値B_R(p,q)に対する論理プロセッサ (p,q)のＣＰＵ利用率の割合、すなわち、

D_R(p,q)=(U_R(p,q)/B_R(p,q))*100　(%)

である。１１４－8 －Rは、走行中の論理プロセッサの統計データ不当インディケータ、１１４－9 －Rは、同リソースキャッピング制御データである。

【００３４】図１１は、タイマ・リクエスト・ブロックTRQB ５０１を表す。これは、ハイパバイザが一定の時刻にタイマ割込みを発生させたいときに、その主記憶内に構成するものである。５０２は、その時刻VALを含む。これは６４ビットあり、ビット５１が １μsecの単位を表す。これは、ホスト実計算機システムの命令プロセッサの所有している現在時刻タイマ(TODタイマという) と同じ形式である。５０３は、割込みリターンアドレスであり、IRA(Interrupt Return Address) として表されている。これは、５０２に示す時刻にタイマ割込みが発生したときに、コントロールを渡すべきアドレスを示している。５０２及び５０３の内容はハイパバイザにより設定される。

【００３５】図１２は、タイマ・リクエスト・キューの構造を表す。これは、ハイパバイザがその主記憶上に構成するソフトウェアのキューである。５１０は、このキューのヘッダであり、５０１－1 はタイマ・リクエスト・ブロックTRQB-1であり、５０２－1 は、そのタイマ要求値、５０３－1 はその割込みリターンアドレスである。さらに、５０１－2 はタイマ・リクエスト・ブロックTRQB-2であり、５０２－2 は、そのタイマ要求値、５０３－2 はその割込みリターンアドレスである。５０１－3 はタイマ・リクエスト・ブロックTRQB-3であり、５０２－3 は、そのタイマ要求値、５０３－3 はその割込みリターンアドレスである。この図では3 個のタイマ・リクエスト・ブロックTRQBしかキューイングされていないが、その数は動的に変動し、空すなわちヘッダだけのときもある。また、タイマ・リクエスト・ブロックTRQBはそのタイマ要求値VALの小さい値から大きな値への順序でキューイングされる。

【００３６】図１３はフリー・タイマ・リクエスト・キューを表す。フリー・タイマ・リクエスト・キューは、フリーなすなわち利用可能なタイマ・リクエスト・ブロックTRQBをキューイングしたものであり、システム立ち上げの時に、ハイパバイザにより作られる。５０１－f －1 は、フリーなタイマ・リクエスト・ブロックTRQB-1である。 ５０１－f －2 は、フリーなタイマ・リクエスト・ブロックTRQB-2である。５０１－f －3 は、フリーなタイマ・リクエスト・ブロックTRQB-3である。このTRQBの個数は動的に変動する。このフリーなTRQBのVAL(５０２－f －1 、５０２－f －2 、５０２－f －3) 及び、IRA(５０３－f －1 、５０３－f －2 、５０３－f －3) は、実際に使用するときに設定されるので任意の値で良い。

【００３７】図１４は、システム制御データを表す。５００は区間の長さΔt(単位はμsec) を含む。これは、図5 に示す１０７実時間の分割におけるひとつの区間の時間的長さである。この区間の時間的長さはシステム立ち上げの時にハイパバイザにより決定されるが、ユーザがシステム・パラメタとして与えることもできる。この５００は、システム立ち上げ時にハイパバイザにより、その主記憶内に構成される。これは、ビット０からビット６３まで６４ビットあり、ビット５１がμsecの単位であり、この形式は、ホスト実計算機システムのTODタイマと同じ形式である。５０１はシステムでのタイムスライスの標準値TS1(μsec単位) である。

【００３８】これは、システム立ち上げの時にハイパバイザにより決定されるが、ユーザがシステム・パラメタとして与えることもできる。この５０１は、システム立ち上げ時にハイパバイザにより、その主記憶内に構成される。これは、ビット０からビット６３まで６４ビットあり、ビット５１がμsecの単位であり、この形式は、実命令プロセッサの CPUタイマと同じ形式である。５５０は、スケジューラタイマ同期用のデータ領域である。５５０－1 は1 ビットのデータK1であり、命令プロセッサー1 のスケジューラ・タイマ設定完了指示データである。５５０－2 (K2)、５５０－3 (K3)、５５０－4 (K4)も同様である。これは、システム立ち上げ時にハイパバイザにより、その主記憶内に、命令プロセッサの台数分確保され、その初期値は0 に設定される。

【００３９】図１５は、システム共通レジスタの現在時

15

刻を表すTODタイマ６００を表す。

【００４０】これは、ビット０からビット６３まで６４ビットあり、ビット５１がμsecの単位である。これは、従来のTODタイマと同じである。

【００４１】図１６は、論理プロセッサタスクの状態遷移図を表す。ハイパバイザは、論理プロセッサタスクをその主記憶上の制御ブロックLGPBで表すことは、図7の１１０に示す通りである。さらに、LGPBは、その中に状態フラグ１１１を持ち、その値Sは、１１５に示すとおりの意味を持つ。図１６において、１２０は、論理プロセッサタスクがレディ (ready)状態であることを示している。そのとき該当のLGPBのS=1である。１２１は、論理プロセッサタスクがウェイト (wait)状態であることを示している。そのとき該当のLGPBのS=3である。１２２は、論理プロセッサタスクがアウトサービス (out-service)状態であることを示している。これは、また、区間におけるサービスを完了している状態、すなわち、該論理プロセッサのＣＰＵ利用率がその制御値に達していることを示している。そのとき該当のLGPBのS=2である。１２３は、論理プロセッサタスクがスリープ (sleep)状態であることを示している。そのとき該当のLGPBのS=0である。矢印１２４は、レディ状態からウェイト状態への状態遷移を表す。他の矢印１２５、１２６、１２７、１２８、１２９も同様にその起点の状態から終点の状態への状態遷移を表す。

【００４２】図１７はレディキュー (Ready Queue) ４１０、アウトサービスキュー (Out Service Queue)４２０、ウィエトキュー (Wait Queue) ４３０からなる。これらは、皆、ハイパバイザによって、その主記憶内に構成されるソフトウェア的な構造体である。２００は、レディキューのヘッダであり、４１０－１、４１０－２、４１０－３、４１０－４　は、レディ状態にある論理プロセッサ制御ブロックLGPB(その状態フラグS=1) であり、順番にキューイングされている。４１０－１が先頭のLGPBである。２０１は、レディキューのヘッダであり、４２０－１、４２０－２、４２０－３　は、アウトサービス状態にある論理プロセッサ制御ブロックLGPB (その状態フラグS=2) であり、順番にキューイングされている。４２０－1が先頭のLGPBである。２０２は、ウェイトキューのヘッダであり、４３０－１、４３０－２、４３０－３、４３０－４　は、ウェイト状態にある論理プロセッサ制御ブロックLGPB(その状態フラグS=3) であり、順番にキューイングされている。４３０－１が先頭のLGPBである。どのキューにおいてもそのキューイングされているLGPBの個数すなわちキューの長さは可変であり、論理プロセッサ実行中に動的に変動する。これらのキューのヘッダはシステム立ち上げ時に、ハイパバイザにより、作られ初期化されるが、キューの実体 (LBPBがキューイングされている構造) は、ハイパバイザのスケジューラにより、作成・管理される。

16

【００４３】図１８は命令プロセッサの所有するレジスタを表す。これらのレジスタは各命令プロセッサ毎にある。３００は、ホストＣＰＵタイマレジスタであり、ビット０からビット６３の６４ビットから成り、ビット５１が1μsecの単位である。これは従来のＣＰＵタイマと同一であり、ビット５１から1μsec毎に1引かれ、ビット０が1即ち、値が負の時割込み(これをＣＰＵタイマ割込みという) がハードウェア的に発生する。３０１は、ゲストＣＰＵタイマレジスタであり、ビット０からビット６３の６４ビットから成り、ビット５１が1μsecの単位である。これは、この命令プロセッサで走行中の論理プロセッサのＣＰＵタイマとして与えられるものであり、従来のＣＰＵタイマと同様の動作を、走行中の論理プロセッサに対して行う。３０２は、ホスト・クロック・コンパレータであり、従来と同一の仕様である。すなわち、ビット０からビット６３の６４ビットから成り、ビット５１が1μsecの単位である。この値は、一定値であり、符号無しの2進数として、図１５の６００の現在時刻TODタイマの値とハードウェアにより、比較される。TODタイマ６００の値が、クロックコンパレータ３０２の値を越えるとハードウェア的に割り込みが発生する。３０３はゲスト汎用レジスタであり、３０４はゲスト制御レジスタであり、３０５はゲストＰＳＷレジスタである。３０５は、走行中の論理プロセッサのＰＳＷとして動作するものであり、実命令プロセッサのＰＳＷと同様の働きを行う。すなわち、ビット０から7は論理プロセッサのシステムマスクSMであり、ビット３２から６３は次に実行する論理プロセッサの命令のアドレス NIA を表す。

【００４４】図１９はモード・インディケータ３０５を表す。３０５は　1　ビットからなり、その値　I　は、３０６に示す意味を持つ。すなわち、I=1 のとき、命令プロセッサは、VMモードであり、I=0 のとき命令プロセッサは、基本モードであることを表す。これは、ハードウェア的なインディケータである。

【００４５】以上の種々のハードウェアリソース、ソフトウェアリソースをベースにして、本発明におけるシステムの動作方法をフローチャートを用いて以下に説明する。

【００４６】図２０に、システムの立ち上げすなわちハイパバイザの立ち上げ時の初期化方法について述べる。ハイパバイザ立ち上げ時、図１３に示すフリーTRQBキューを構成する。その主記憶内にTRQBを構成しヘッダ５２０からキューイングする (７２０) 。TRQBの個数は最大サポートＶＭ台数＊(ＶＭ当たりの最大論理プロセッサ台数) 分用意する。例えば、１０台のＶＭと各ＶＭについて論理プロセッサの台数１０台であれば、１０＊１０＝１００個のTRQBを用意して、キューイングしておく。各TRQBの内容は0 クリアしておく。　次に図１４に示すシステム・タイム・スライスTS1を計算し、システムの

領域５０１に格納する（７２１）。システム・タイム・スライスは、いろいろな決定方法があるが、ここでは、1,000Kステップだけ単純命令（例えばLoad命令）を実行し、その実行時間をシステム・タイム・スライスTS1（単位はμsec）とすることにする。次に図１４に示す実時間を分割する区間の長さΔt（単位はμsec）を決定し、システム領域５００に格納する（７２２）。Δtの値は、ここでは1秒すなわち、1,000,000μsecとするが、この値は、ユーザ指定のシステム・パラメタにより変更可能とする。次に図１２に示すタイマリクエストキューのヘッダ５１０を構成し、それを初期化し、キューの中に何もキューイングされていない状態、すなわち、空の状態にする（７２３）。

【００４７】次に図１４に示すスケジューラタイマ同期データ領域５５０を0クリアする。すなわち、５５０－１のK1、５５０－２のK2、５５０－３のK3、５５０－４のK4を0クリアする。この例では4台の命令プロセッサの場合であるが、N台の場合は、K1,K2,･･･,KNまでのデータを0クリアする（７２４）。　次に、このデータを用いてスケジューラタイマの同期処理を行う（７２５）。これで、立ち上げ時の初期化を終える。

【００４８】この７２５の処理について、図２１に詳述する。　先ず、図１３のフリーTRQBキューの先頭のTRQBをキューより取り外し、次のTRQBを先頭とする。該TRQBのタイマ要求値（５０２相当）に以下の式によって計算されるVAL0を格納する（７２５－１）。

【００４９】

VAL0 = 現在時刻TOD(600) + 区間の長さΔt(500)

該TRQBの割込みリターン・アドレスIRAにスケジューラのある入りロアドレス（E0）を設定する（７２５－２）。この入りロアドレス（E0）は、ハイパバイザによって決められるアドレスである。該TRQBを図１２のタイマ・リクエスト・キューにキューイングする（７２５－３）。このときタイマ・リクエスト・キューは空の状態であるから、このTRQBは、先頭にキューイングされる。次に該TRQBの上記のタイマ要求値VAL0を現命令プロセッサのホスト・クロック・コンパレータ・レジスタ３０２に設定する。さらに、現命令プロセッサの番号をiとするとき、

５５０-i(値Ki)＝1

とする（７２５－４）。これにより、現命令プロセッサについては、現在時刻からΔtμsec後に、ホスト・クロック・コンパレータの割込みが発生するようになる。次に他の命令プロセッサにも同様のタイマを設定するために上記のVAL0をパラメタとしてプロセッサ間スケジューラタイマ同期処理を行う（７３０）。以上でスケジューラタイマ同期処理を終える。

【００５０】この７３０の処理を図２２に詳述する。すなわち、タイマ要求値VAL0をパラメタとして、自分以外の全ての命令プロセッサに対して外部割込み要求を発行する。このとき、これが、スケジューラタイマ同期要求であることを示すパラメタも、相手命令プロセッサへ渡す（７３０－２）。次にスケジューラタイマ同期データ領域５５０の全てのK1,K2,････,KN　について、K1=1　かつ　K2=1　かつ･･･かつ　KN=1　であるかをチェックする（７３０－３）。これらのデータの初期値は0であり、命令プロセッサ　(番号i)　が、タイマ要求値VAL0に対して、そのクロック・コンパレータの設定を完了したとき、すなわち、図２１に示す７２５－4 相当の処理を完了したとき、５５０のki=1とする。したがって、７３０－3は、全ての命令プロセッサが、そのクロック・コンパレータに対して上記のタイマ要求値VAL0を設定するのを待つものである。この完了後図２０の７２６へ行く（すなわち、ハイパバイザの立ち上げ処理を終える）。

【００５１】図２３に外部割り込み処理を示す。図２２に示すプロセッサ間スケジューラタイマ同期処理で、各プロセッサにスケジューラタイマ同期要求の外部割り込み要求を発行する（７３０－２）と、各プロセッサに外部割り込みが発生する。図２３は、一般的な外部割り込みの処理を示す。先ず、外部割り込みの種類を判断する。すなわち、スケジューラタイマ同期要求の外部割り込みかを判断する（７４０）。これは、従来どおり、送られてきた割り込み要求コードから判断できる。そうでないときは、クロック・コンパレータ割り込みかどうかを判断する（７４０－１）。そのときは、クロック・コンパレータ割り込みの処理を行うために９１０－1へ行く。そうでないときは、従来通り、該当の外部割り込み処理を行い、一般的なスケジューラ９９９へ行く（７４１）。そうであるときは、渡されたパラメタVAL0を現プロセッサのホストクロックコンパレータ３０２に設定する（７４２）。これにより、ほぼ、時刻VAL0に、現命令プロセッサのホスト・クロック・コンパレータの割込みが発生するようになる。このタイマ割込み要求が、全ての命令プロセッサにおいて設定されたことを確認するために、スケジューラ・タイマ同期データ領域５５０の全てのK1,K2,･･･,KN　について、K1=1 かつ　K2=1 かつ･･･かつ KN=1であることをチェックする。そうでないときは、くりかえし、同条件をチェックする。そうなったときに、ループから抜けて、一般的なスケジューラ９９９へ行く（７４４）。

【００５２】これにより、全命令プロセッサにおいて、ほぼ、時刻VAL0にホスト・クロック・コンパレータの割込みが発生することになる。

【００５３】ユーザがＶＭをアクティベイト（起動）すると、図２４に示すようにＶＭ下の全ての命令プロセッサのタスク制御ブロックLGPBを初期化する（７００）。この初期化処理を図２５に詳述する。起動したＶＭの番号をp（p=1,2,･･･のいづれかの値）とする。ＶＭp下の各論理プロセッサ制御ブロックLGPB(p,q)　(q=1,2,3,･･･）について、以下の初期化を行う。　各LGPBは、図7

19

に示す論理プロセッサ・タスク制御ブロック１１０の構造を持つ。すなわち、タイムスライス初期値　TS0(p,q)を計算し、該LGPBの１１４－1のところに格納する。このTS0(p,q)は、システム・タイム・スライス標準値TS1（５０１）を、そのまま持って来ても良い。次に残りのタイムスライスTS(p,q)に、TS0(p,q)を初期値として設定し、114-2に格納する。次に区間CPUサービス量データC(p,q)を0に初期化して１１４－3に格納する。初期制御値A(p,q)を計算し、１１４－4に格納する。この計算方法は、図4、図6で説明したとおりアクティブなＶＭの間のサービス率に基づいて計算される。次に制御値B(p,q)にA(p,q)を初期値として設定し、１１４－5に格納する。次に区間CPU利用率U(p,q)を0に初期化する。次に、区間CPU充足度D(p,q)を0に初期化する。さらに統計データ不当インディケータ１１４－8　ｓｉｎｖ（ｐ，ｑ）=0とする。これは、上記の１１４－2（TS(p,q)）、１１４－3（C(p,q)）、１１４－6(U(p,q))、１１４－7(D(p,q))が最新のデータであることを表す。VMp下のすべての論理プロセッサの制御ブロックLGPB(p,q)について、以上の初期化を終えたら７００－3へ行き、ＶＭのアクティベイト処理を終える。

【００５４】図２６の説明。ＶＭのアクティベイト処理を完了すると、そのＶＭ上でオペレーティング・システム(OS)を起動する。そうすると、ＶＭ下の全ての論理プロセッサがスタートされてレディキュー４１０にキューイングされ、コントロールが図２６に示すディスパッチ処理に渡される。このディスパッチ処理では、先ず、図１７のレディキュー４１０の先頭のタスク制御ブロックLGPBをキューからはずして、取り出す。そのLGPBの１１２の場所からＶＭ番号p，論理プロセッサ番号qを得る（７１３－2）。次に、該LGPBの残りのタイム・スライス・データTS(p,q)を場所１１４－2から読み出して、現命令プロセッサのホストCPUタイマ・レジスタ300に設定する（７１３－3）。これにより、もし、このTS(p,q)を該論理プロセッサが使い尽くしたときには、ホストCPUタイマの割込みが発生する。これにより、別の論理プロセッサに制御を移すことができるようになる。次に統計データ不当インディケータsinv(p,q)=1とする（７１３－4）。これは、１１４－2（TS(p,q)）、１１４－3（C(p,q)）、１１４－6(U(p,q))、１１４－7(D(p,q))が古いデータである、すなわち、最新のデータではないことを表す。これは、これからまさに該当論理プロセッサ(p,q)をディスパッチしようとしており、ディスパッチ後はこれらのデータは最新でなくなるからである。さらに、該LGPB内の論理プロセッサリソースデータ１１３を現在の命令プロセッサのレジスタに設定し、該論理プロセッサを、現命令プロセッサ上で走行させる。すなわち、該論理プロセッサのLGPB内の汎用レジスタデータ１１３－1をゲスト汎用レジスタ３０３に設定し、制御レジスタデータ１１３－2をゲスト制御レジスタ３０４に設定し、PSWデータ１１３－3をゲストPSWレジスタ３０５に設定し、CPUタイマ・データ１１３－4をゲストCPUタイマ３０１に設定し、動作モードをVMモードにする。即ち、モードレジスタ３０５のI=1とし、さらに、該LGPBのアドレスを、走行中論理プロセッサ表示レジスタ３０７に設定し、コントロールをゲストPSWレジスタ３０５が示すところへ渡す。以後命令プロセッサは、ゲストPSWレジスタ３０５、及びゲスト制御レジスタ３０４によって制御される。すなわち、該当の論理プロセッサを現命令プロセッサ上で走行させることになる（７１３－5）。以上によってディスパッチ処理を終える。

【００５５】図２７、２８、２９の説明:　このようにして、論理プロセッサが実の命令プロセッサ上で走行させられる。走行中の論理プロセッサは、いろいろな要因で中断される。そのひとつは、ホストの割込みである。たとえば、ホストのクロック・コンパレータの割込みや、ホストのCPUタイマの割込みがそうである。ホスト・クロック・コンパレータ割込みは、図１８に示す、ホスト・クロック・コンパレータ・レジスタ　３０２の値が、図１５に示すシステム共通レジスタ６００の現在時刻TODの値を越えたときに発生する。これは、従来通りである。ホストＣＰＵタイマの割込みは、図１８に示すホストＣＰＵタイマ・レジスタ３００の値が負になったときに発生する。これも従来通りである。図２１に示すスケジューラタイマ同期処理により、全命令プロセッサで同時にスケジューラタイマ用のホスト・クロック・コンパレータの割込みが発生する。また、図２６に示す、ディスパッチ処理でホストＣＰＵタイマ・レジスタ３００に残りのタイム・スライスTS(p,q)（論理プロセッサ(p,q)用のタイムスライス）を設定したことにより（７１３－3）、該当の論理プロセッサが、そのタイム・スライスを使い切るとホストＣＰＵタイマ割込みが発生する。これを、タイムスライスエンドの割込みという。これらの制御は、従来通りである。これらのホスト割込み処理の後、図２７、２８、２９に示す論理プロセッサの中断処理が呼ばれる。ここでは、先ず、走行中であった論理プロセッサの制御ブロックLGPBのアドレスをレジスタ３０７から求める。今それをLGPB_Rとする（８００－2）。このLGPB_Rから現ＶＭ番号p,論理プロセッサ番号q　を求める（８００－3）。これは、LGPB_Rの場所１１２から読み出してくればよい（図7参照）。次に、LGPB_Rの統計データ不当インディケータsinv(p,q)=1であるかを判断する。そうである場合は、該LGPB_R内の種々の統計データを更新する必要があるので、８００－4へ行く。そうでない場合は、すでに更新済みであるので、ただちにリターンする。次に、該論理プロセッサのＣＰＵ消費時間を計算するために、現命令プロセッサのホストＣＰＵタイマレジスタ３００の値を読み出す。今、その値をTS_R'とする（８００－4）。次に該LGPB_Rの残

20

21

りのタイムスライスデータを１１４－2 －Rから読み出す。今、その値をTS_Rとする（８００－5）。これは、該論理プロセッサのディスパッチ時のホストＣＰＵタイマの値がTS_Rμsecであり、それが1μsecごとにビット５１から1が減算されて行き（図18の３００参照）、今処理している論理プロセッサの中断時にTS_R'μsecとなったことを意味する。したがって、該論理プロセッサのＣＰＵ消費時間 u μsecは、

u = (TS_R) - (TS_R')

によって求められる（８００－6）。このuμsecは、最後のディスパッチから、最初の中断までの論理プロセッサの消費したＣＰＵ時間である。次に、該論理プロセッサ(p,q)の累計のＣＰＵ消費時間を計算する。これは、以下の式による。

【００５６】すなわち、LGPB_Rの場所１１４－3 －Rの値C(p,q)_Rを読み出す。これは、該論理プロセッサの区間におけるＣＰＵ消費時間の累計を表す。そこで、

C(p,q)_R = C(p,q)_R + u

とし、更新されたC(p,q)_Rの値を該LGPB_Rの１１４－3 －Rに格納する（８００－7）。次に、該論理プロセッサの区間におけるＣＰＵ利用率U(p,q)_R (%) を求める。システムデータ領域５００から区間の長さΔtμsecを得る。さらに、８００－7 で更新したC(p,q)_R(これは、区間における該論理プロセッサのＣＰＵ消費時間) を用いて、

U(p,q)_R = (C(p,q)_R/Δt)*100　(%)

これを、該論理プロセッサの区間ＣＰＵ利用率として、該LGPB_Rの１１４－6 －Rに格納する（８００－8）。次に、この区間ＣＰＵ利用率が制御値を越えるかどうかを判断する。すなわち、８００－8 で計算した区間ＣＰＵ利用率U(p,q)_R(これは、該論理プロセッサのＣＰＵ利用率) と、該LGPB_Rの場所１１４－5 －Rに格納されている制御値B(p,q)_R(これも該論理プロセッサに関する値) を用いて、

(U(p,q)_R) < (B(p,q)_R)

かどうかを判断する（８００－9）。もし、そうであるならば、まだ、現区間においては、制御値に達していないので、再びレディキューにキューイングしてディスパッチするために８００－１０へ行く。もし、そうでないときは、すでに制御値に達したのでアウトサービスキューへキューイングするために８００－１２へ行く。先ず、８００－１０から説明する。ここでは、区間における該論理プロセッサ(p,q)のＣＰＵ充足度(D(p,q)_R (%))を計算する。

【００５７】

D(p,q)_R = ((U(p,q)_R)/(B(p,q)_R))*100

この値を該論理プロセッサ(p,q)の区間ＣＰＵ充足度として該LGPB_Rの場所１１４－7 －Rに格納する（８００－１０）。次に、該LGPB_Rをレディキューにキューイングするために９００へ行く。９００では、レディキュー

22

４１０に該LGPB_Rをキューイングするために、そのヘッダ２００から検索する。このレディキューは、区間ＣＰＵ充足度D(p,q)の小から大へソートしてキューイングしているので、該LGPB_Rの区間ＣＰＵ充足度 D(p,q)_R についてソートし適切な位置にキューイングする（９００）。レディキューは先頭のLGPBからディスパッチされるので、このソートにより、各論理プロセッサの区間ＣＰＵ充足度が小さいものから先にディスパッチされることになり、従って、各論理プロセッサ(p,q)の現在の制御値B(p,q)に比例したＣＰＵサービスが行われるようになる。９００の処理完了の後は、該LGPB_Rの以上の統計データの更新を終えたので、その統計データ不当インディケータ 114-8 の値sinv(p,q)=0とする（８００－１１）。これで中断処理を終えて、呼び出し元(caller) へもどる。次に、区間ＣＰＵ利用率が制御値に達したときの処理を８００－１２から説明する。ここでは、該LGPB_Rをアウトサービスキュー４２０のヘッダ２０１を検索し、該LGPB_Rをキューの最後へキューイングする（８００－１２）。この完了後８００－１１へ行き、該論理プロセッサの統計データ不当インディケータ 114-8 の値sinv(p,q)=0とする。これにより、中断処理を終え、呼び出しもとへリターンする。

【００５８】図３０、３１にレディキューへのキューイング方法を詳述する。これは９００の処理の詳細である。先ず９００－2 では、レディキュー４１０をそのヘッダ２００より検索し、先頭のLGPBにアクセスし、これをカレントLGPBとする。キューイングすべき論理プロセッサ制御ブロックLGPBを該当LGPBとする（９００－２）。該当LGPBの区間CPU充足度(114-7相当部分の値) をD(p,q)_Gとする。カレントLGPBの区間ＣＰＵ充足度（114-7相当部分の値）をD(p,q)_Cとする。そこで、

(D(p,q)_G) ≧ (D(p,q)_C)

かどうかを判断する（９００_3）。そうであるときは、まだ、後方へキューイングする必要があるのであるので、キュー内の次のLGPBが存在するかを判断する（９００－4）。あればそれをカレントLGPBとする（９００－5）。その後、再び９００－3 へ行く。９００－3 の条件が成立しないときは、カレント位置の直前に該LGPBをキューイングする（９００－6）。それでキューイングは完了する。

【００５９】９００－4 の判断でキュー内の次のLGPBが存在しないときは、９００－7 に行き、そこで、レディーキューの最後にキューイングする（９００－7）。これにより、キューイングを完了する。

【００６０】図３２、３３にクロック・コンパレータ割り込みの処理を述べる。クロック・コンパレータ割り込みは外部割り込みの一種であり、図２３に示す外部割り込み処理でクロック・コンパレータ割り込みと判断されると９１０－1 に制御が来る。９１０－2 で、図１２のタイマ・リクエスト・キューのヘッダ５１０より、その

先頭のTRQBにアクセスする。もし、タイマ・リクエスト・キューが空のときは、９１０－8 へ行く。該タイマ・リクエスト・キューが空でないときは先頭のTRQBにアクセスする。そのTRQBのタイマ要求値VAL(５０２) と現在時刻TOD(６００) を比較する (９１０－3) 。もし、

VAL < TOD

ならば、該TRQBをタイマ・リクエスト・キューより取り外す。これを該当TRQBとする (９１０－4) 。かかるTRQBが無いときは、９１０－8 へ行く。かかるTRQBが有るときは、次のTRQBを先頭のTRQBとするようにヘッダ５１０を更新し (９１０－4) 、９１０－5 へ行く。９１０－5 においては、該当TRQBが、スケジューラ・タイマ割込み用のものであるかを判断する。それは、その割込みリターンアドレスIRAがスケジューラの特定の入り口 E0 であるかを判断すればよい。特定のアドレスE0を入り口とするスケジューラの処理においては、スケジューラタイマの割込み処理を行う。これは、図３４で説明する。スケジューラ・タイマ割込み用のものである場合は、次の区間Δtμsec後のスケジューラタイマの割込みを発生させるために、該TRQBのタイマ要求値VAL(502)に以下の値を設定する (９１０－6) 。

【００６１】VAL= 現在時刻TOD (600) + Δt(500)

その後、再び、該TRQBを図１２のタイマ・リクエスト・キューにキューイングする。このタイマ・リクエスト・キューにおいては、各TRQBのタイマ要求値VALの小さい値から大きい値へとソートしてキューイングする (９１０－7) 。このあと９１０－8 へ行く。９１０－5 でスケジューラ・タイマ割込み用のものでないときは、９１０－8 へ行く。９１０－8 では、あらたにタイマ・リクエスト・キューが非空かを判断し、非空のときは、その新しい先頭のTRQBのタイマ要求値(５０２相当部分の値) VALをホスト・クロック・コンパレータ・レジスタ３０２に設定し (９１０－9) 、９１０－１１へ行く。かかる新しい先頭のTRQBが存在しないとき、すなわち、タイマ・リクエスト・キューが空になったときは、現時刻に500ミリ秒程度を加算した値を現命令プロセッサのホスト・クロック・コンパレータに設定する (９１０－１０) 。このあと走行中論理プロセッサの中断処理を行うために９１０－１１へ行く。９１０－１１では、図２７の処理(走行中論理プロセッサの中断処理) をコールする。その後９１０－１２へ行く。ここでは、９１０－3と９１０－4 で見つけた該当TRQBが存在するときは、その割り込みリターンアドレス (５０３相当部分の値 IRAが、このリターンアドレスを表す) へコントロールを渡す (９１０－１３) 。かかるTRQBが存在しないときは、一般的なスケジューラ９９９へ行く (９１０－１４) 。

【００６２】図３４の説明。図３４は、スケジューラの特定の入り口E0からの処理を表す。ここでは、スケジューラ・タイマ・割込み処理を行う。ここには、スケジューラ・タイマ用のTRQBの割込みリターンアドレスIRAからコントロールが渡ってくる (図３３の９１０－１３参照) 。このスケジューラ・タイマ割込みは、図２１に示すスケジューラタイマ同期処理及び図３２に示すホスト・クロック・コンパレータ割込み処理での各区間Δt経過後の割込み要求の設定 (９１０－6 、９１０－7) によって、各命令プロセッサでΔtμsec後に同期して発生するようになる。各命令プロセッサは、スケジューラタイマ割込みによって図３４に示す処理を行うようになる。すなわち、先ず、レディキュー４１０のスケジューラタイマ割込み処理を行う (９２０) 。そのあと、アウトサービスキュー４２０のスケジューラタイマ割込み処理を行う (９３０) 。そのあと、ウェイトキュー４３０のスケジューラタイマ割込み処理を行う (９４０) 。この後一般的なスケジューラの処理９９９へ行く。

【００６３】図３５の説明: 図３５は、レディキューに対するスケジューラタイマ割込み処理、すなわち、９２０の処理の詳細である。先ず、図１７のレディキュー４１０をヘッダ２００より、検索し、レディキューが非空であるかを判断し、非空のときは、その先頭のLGPBを該当のLGPBとする。その該当のLGPBのＶＭ番号をp,論理プロセッサ番号をqとする (９２０－1) 。その該当LGPBが無いときは次の処理９３０へ行く。その該当LGPBが存在するときは、９２０－4 へ行き、該LGPBについて現区間での終了処理を行う。９２０－4 では先ず該LGPBの残りのタイムスライスTS(p,q) (１１４－2) をタイムスライス初期値TS0(p,q) (１１４－1) に初期化する。これは、次の区間Δtにおいては、タイムスライス初期値から始めれば良いからである。次に該当LGPBの区間ＣＰＵサービス量C(p,q)を0クリアする。これも次の区間に向けての初期化である。次に、区間におけるＣＰＵサービス量の制御値B(p,q)を更新する。この制御値の更新を行うことが本特許の特徴的処理である。すなわち、該LGPBのリソースキャッピング制御データ (１１４－9) rc=2、すなわち、大域的リソースキャッピング指定のとき、該当LGPBの現在の制御値B(p,q) (１１４－5) 、その初期制御値A(p,q) (１１４－4) 、区間ＣＰＵ利用率をU(p,q) (１１４－6) とするとき、

B(p,q) = A(p,q) + (B(p,q)-U(p,q))

とする。この更新されたB(p,q)を該当LGPBの114-5に格納する。これは、区間における未使用の部分 (B(p,q)-U(p,q)) だけ、初期制御値A(p,q)に加算することにより、大極的に見て、論理プロセッサ (p,q)のＣＰＵ利用率をA(p,q)以下に抑制するためである。上記のrc=0or1のときは、リソースキャッピングの指定無しか、局所的なリソースキャッピングの指定であるのでB(p,q)の値は常にA(p,q)に等しくする。これにより、制御値の更新が終わったので、上記区間ＣＰＵ利用率U(p,q)を0クリアする。次に該当LGPBの区間ＣＰＵ充足度D(p,q) (１１４－

25

7）もoクリアする。これも次の区間に向けての初期化である。以上で該LGPBの次の区間Δtに向けての初期化を終える。次にレディキュー上に次のLGPBがキューイングされているかを判断し、あればそれを新たな該当LGPBとし（９２０－５）、９２０－２へ行く。ここで、再び新たな該当LGPBが有るかどうかを判定し、有れば、再び、新たな該当LGPBの初期化のために９２０－４へ行く。なければ、以上でレディキューに対するスケジューラタイマ処理を終えて、次の処理９３０へ行く。

【００６４】図３６の説明:　図３６は、図３４の処理９３０の詳細である。先ず、図１７のアウトサービスキュー４２０をヘッダ２０１より検索し（９３０－１）、アウト・サービス・キューが非空であるかを判断し、非空のときは、その先頭のLGPBを該当のLGPBとする。その該当のLGPBのＶＭ番号をp,論理プロセッサ番号をqとする（９３０－２）。その該当LGPBが無いときは次の処理９４０へ行く。その該当LGPBが存在するときは、９３０－４へ行き、該LGPBについて現区間での終了処理を行う。これは、図３１の９２０－４と同じ処理である（９３０－４）。この後該LGPBをレディキュー４１０にキューイングする。すなわち、そのヘッダ２００から検索して最後の位置にキューイングする（９３０－５）。これは、次の区間Δtにおいて、アウトサービスキューに有る全ての論理プロセッサ(p,q)を再び、新制御値B(p,q)に基づいて、サービスするためである。次にアウトサービスキュー４２０上に次のLGPBがキューイングされているかを判断し、あればそれを新たな該当LGPBとし（９３０－６）、９３０－３へ行く。ここで、再び新たな該当LGPBが有るかどうかを判定し、有れば、再び、新たな該当LGPBの初期化のために９３０－４へ行く。なければ、以上でアウトサービスキューに対するスケジューラタイマ処理を終えて、次の処理９４０へ行く。

【００６５】図３７の説明:　図３７は、図３４の処理９４０の詳細である。先ず、図１７のウェイトキュー(wait queue)４３０をヘッダ２０２より検索し（９４０－１）、ウェイトキューが非空であるかを判断し、非空のときは、その先頭のLGPBを該当のLGPBとする。その該当のLGPBのＶＭ番号をp,論理プロセッサ番号をqとする（９４０－１）。その該当LGPBが無いときは、処理完了なので一般的なスケジューラ９９９へ行く。その該当LGPBが存在するときは、９４０－３へ行き、該LGPBについて現区間での終了処理を行う。これは、図３５の９２０－４と同じ処理である（９４０－３）。次にウェイトキュー４３０上に次のLGPBがキューイングされているかを判断し、あればそれを新たな該当LGPBとし（９４０－４）、９４０－２へ行く。ここで、再び新たな該当LGPBが有るかどうかを判定し、有れば、再び、新たな該当LGPBの初期化のために９４０－３へ行く。なければ、以上でウェイト・キューに対するスケジューラタイマ処理を終えて、一般的なスケジューラ９９９へ行く。

26

【００６６】図３８は一般的なスケジューラの処理である。ここでは、種々のシステムアクティビティの処理を行ったり、統計量を採ったりする。さらにレディキューを検索し、走行可能な論理プロセッサがあればそれをディスパッチしたりする。レディキューが空のときは、アウトサービスキューを検索し、リソースキャッピングの指定のないLGPB、すなわち、そのLGPBのリソースキャッピング制御データ（114-9）rc=0のLGPBだけをレディキューに上げてディスパッチする。この処理は従来の一般的なスケジューラの処理と同じであり、詳細は省略する。

【００６７】以上の実現方法によって大域的なリソースキャッピングを実現することができるが、さらに、この実現方式は、従来の局所的なリソースキャッピングも容易に実現することができる。それは、図３５の９２０－４の処理において、該当LGPB(p,q)の制御値B(p,q)の更新を止めていつも初期値A(p,q)とすれば、各区間においてＣＰＵサービス量U(p,q)が、いつも初期値A(p,q)によって抑制されることになる。これを該当のＶＭの全論理プロセッサについて行えば、そのＶＭに対して、局所的なリソースキャッピングを実現することができる。これをＶＭ毎に制御する、すなわち、あるＶＭは大域的なリソースキャッピング制御を行い、あるＶＭは局所的なリソースキャッピング制御を行うと云うことも容易である。

【００６８】最後に図1のブロック図と、フローチャートとの対応関係を示す。図1のシステム初期化４００は、図２０のハイパバイザ立ち上げ時の初期化処理を全て含む。

【００６９】周期タイマ割込み発生部４０３は、図２１のスケジューラタイマ同期処理、図２２のプロセッサ間スケジューラタイマ同期処理、図３２、３３のホストクロック・コンパレータ割込み処理を全て含む。スケジューラ４０２は、図２３の外部割込み処理、図２４、図２５のＶＭのアクティベイト処理、図２７、２８、２９の論理プロセッサ中断処理、図３０、３１のレディキューへのキューイング処理、図３４のスケジューラタイマ割込み処理、図３５、図３６、図３７のレディキュー、アウトサービスキュー、ウェイトキューに対するスケジューラタイマ割込み処理、図３８の一般的なスケジューラの処理の全てを含む。ディスパッチャ４０１は、図２６のディスパッチ処理を全て含む。４０４サービス量（Ｕ）計算部分は、図２７、２８、２９の論理プロセッサの中断処理に相当する。４０５のキュー操作部分は、図３０、３１のレディキューへのキューイング処理を全て含む。４０６の制御値(B)設定部分は、図３５、図３６、図３７のレディキュー、アウトサービスキュー、ウェイトキューに対するスケジューラタイマ割込み処理に相当する。

【００７０】

27

【 発明の効果】本特許により、仮想計算機に対するＣＰＵサービス量をユーザの指定量に抑えるというリソースキャッピングの目的において、局所的のみならず大域的に、その目的を実現することができる。仮想計算機毎に局所的または大域的の指定を行うことができる。これにより、ユーザの意図にそったリソースキャッピングを実現することができる。

【 図面の簡単な説明】

【 図1 】本特許の概要を表すハイパバイザのブロック、およびコントロールやデータの流れを表す図である。

【 図2 】ホスト 実計算機システムを表す図である。

【 図3 】仮想計算機の構成図である。

【 図4 】仮想計算機ＶＭの定義画面を表す図である。

【 図5 】実時間の分割方法を示す図である。

【 図6 】記号の説明を表す図である。

【 図7 】論理プロセッサ・タスク制御ブロックの構造を表す図である。

【 図8 】論理プロセッサ (p,q)の制御データを表す図である。

【 図9 】走行中の論理プロセッサ (p,q)の制御データを表す図である。

【 図1 0 】走行中の論理プロセッサ指示レジスタを表す図である。

【 図1 1 】タイマ・リクエスト・ブロック (TRQB)の構造を表す図である。

【 図1 2 】タイマ・リクエスト・キューを表す図である。

【 図1 3 】フリーTRQBのキューを表す図である。

【 図1 4 】システムにおける主記憶上の共通データを表す図である。

【 図1 5 】システムにおける共通レジスタを表す図である。

【 図1 6 】論理プロセッサタスクの状態遷移図である。

【 図1 7 】レディキュー(ready queue) ,アウトサービスキュー(out servicequeue) ,ウェイトキュー(wait queue) を表す図である。

【 図1 8 】命令プロセッサのレジスタ類を表す図である。

【 図1 9 】モードレジスタを表す図である。

【 図2 0 】ハイパバイザ立ち上げ時の初期化処理のフローチャート である。

【 図2 1 】スケジューラタイマに対する同期処理のフローチャート である。

【 図2 2 】命令プロセッサ間のスケジューラタイマに関する同期処理を表すフローチャート である。

【 図2 3 】外部割込み処理を表すフローチャート である。

28

【 図2 4 】ＶＭのアクティベイト 処理のフローチャート である。

【 図2 5 】ＶＭのアクティベイト 処理のフローチャート である。

【 図2 6 】論理プロセッサに対するディスパッチ処理のフローチャート である。

【 図2 7 】論理プロセッサの中断処理のフローチャート (1/3)である。

【 図2 8 】論理プロセッサの中断処理のフローチャート (2/3)である。

【 図2 9 】論理プロセッサの中断処理のフローチャート (3/3)である。

【 図3 0 】レディキューへのキューイング処理のフローチャート (1/2)である。

【 図3 1 】レディキューへのキューイング処理のフローチャート (2/2)である。

【 図3 2 】ホストクロック・コンパレータ割込み処理のフローチャート (1/2)である。

【 図3 3 】ホストクロック・コンパレータ割込み処理のフローチャート (2/2)である。

【 図3 4 】スケジューラタイマ割込み処理のフローチャート である。

【 図3 5 】レディキューに対するスケジューラタイマ割込み処理のフローチャート である。

【 図3 6 】アウトサービスキューに対するスケジューラタイマ割込み処理のフローチャート である。

【 図3 7 】ウェイトキューに対するスケジューラタイマ割込み処理のフローチャート である。

【 図3 8 】一般的なスケジューラの処理を表す図である。

【 図3 9 】あるＶＭのある負荷に対する本来のＣＰＵ利用率を実時間の経過とともに表す図である。

【 図4 0 】図3 9 のＶＭに対して局所的リソースキャッピングを適用したときのＣＰＵ利用率を表す図である。

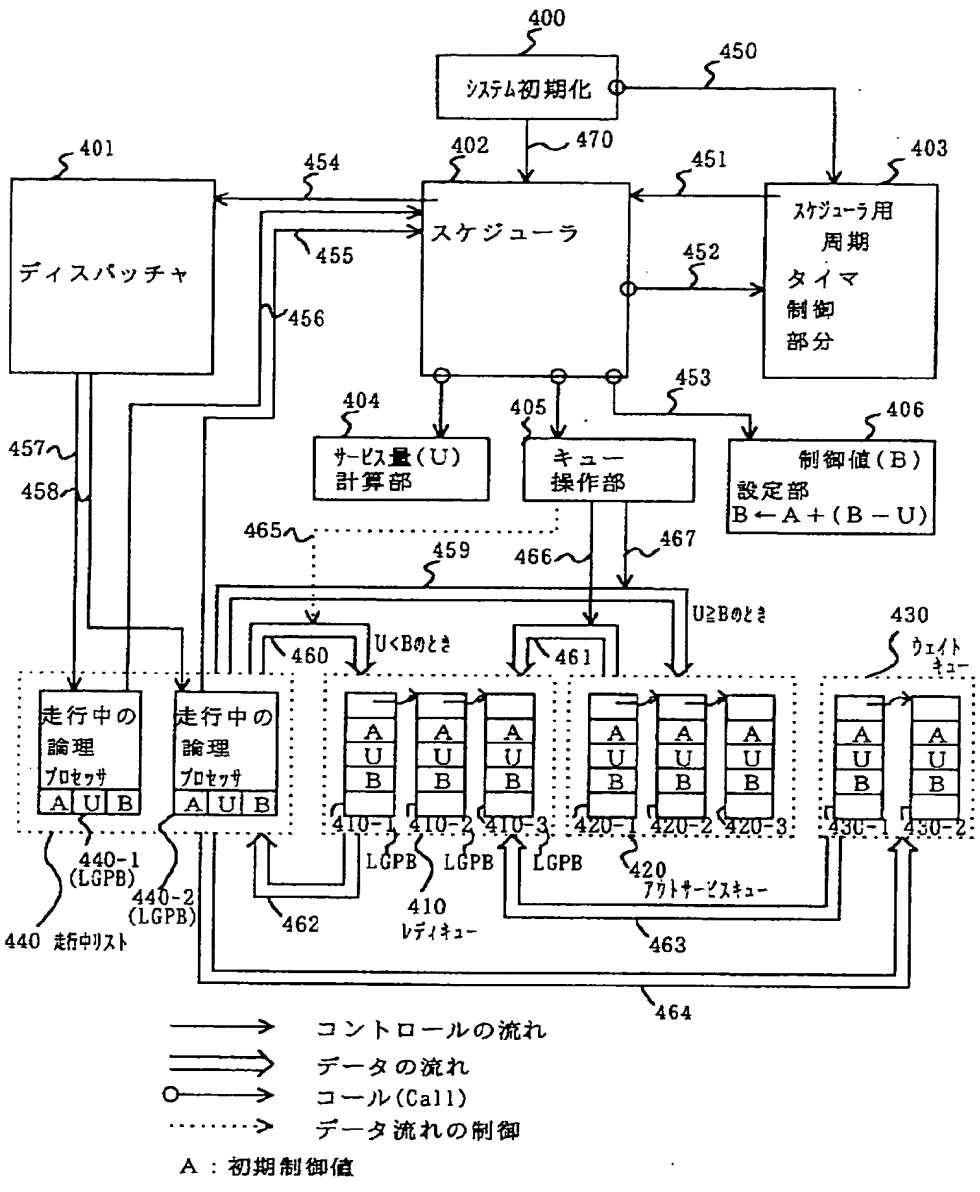
【 図4 1 】図3 9 のＶＭに対して大域的リソースキャッピングを適用したときのＣＰＵ利用率を表す図である。

【 符号の説明】

400･･･システム初期化処理の部分、401･･･ディスパッチャ、402･･･スケジューラ、４０３・・・スケジューラ用周期タイマ制御部分、４０４・・・サービス量(Ｕ) 計算部分、４０５・・・キュー操作部分、４０６・・・制御値(Ｂ) 設定部分、４４０・・・走行中の論理プロセッサのリスト、４１０・・・レディキュー(Ready queue) 、４２０・・・アウトサービスキュー(Out service queue) 、４３０・・・ウェイトキュー(Wait queue) 。

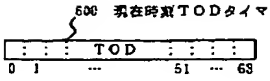
【 図1 】

図 1



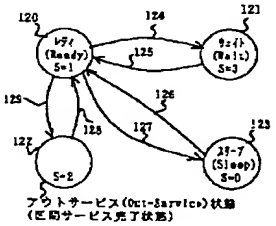
【 図15 】

図15
システム共通レジスタ
600 現在時刻TODタイマ



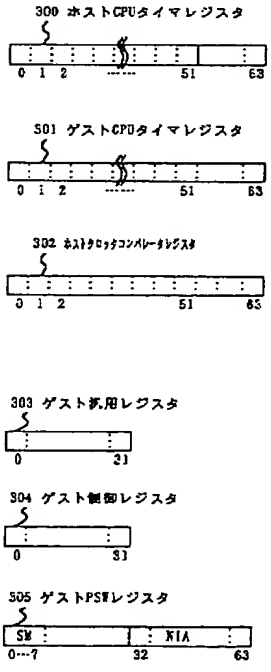
【 図16 】

図16
論理プロセッサタスク状態遷移



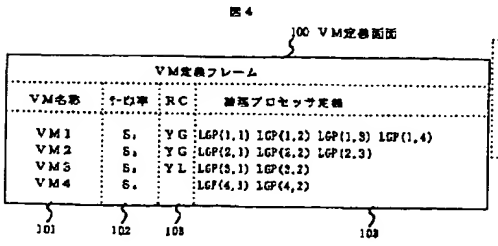
【 図18 】

図18
命令プロセッサのレジスタ群



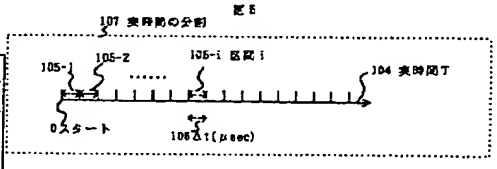
【 図4 】

図4



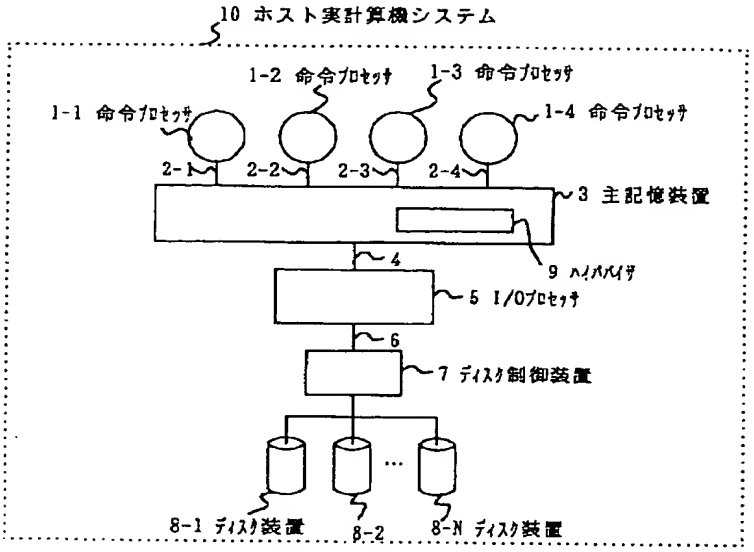
【 図5 】

図5



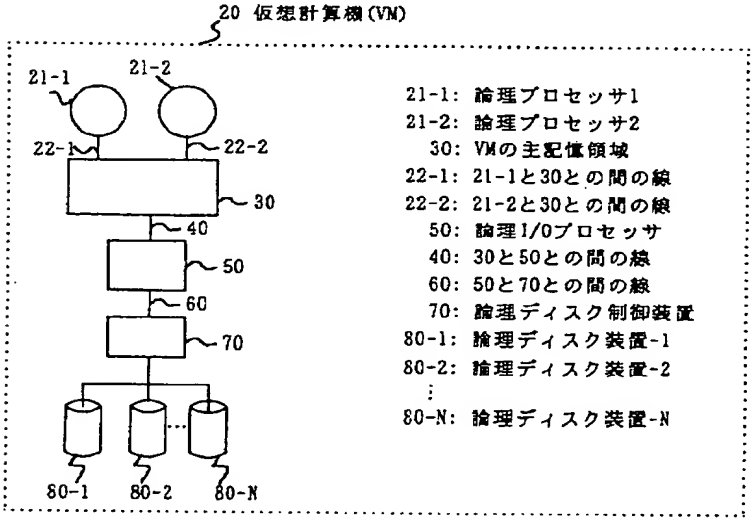
【 図2 】

図 2



【 図3 】

図 3



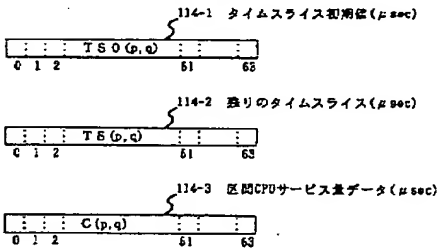
【 図6 】

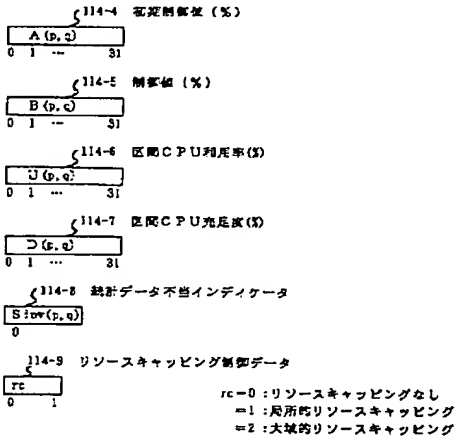
図6
記号説明

記　号	意　　味
C(p,q)	区間における論理プロセッサLGP(p,q)のCPU busy time (μsec)
Δt	区間の時間的長さ：どの区間も同じ大きさとする。(μsec)
A1 A2 ⋮ AN	VM1のCPU利用率の初期制御値（%） VM2のCPU利用率の初期制御値（%） ⋮ VMNのCPU利用率の初期制御値（%）
A(p,c)	論理プロセッサLGP(p,c)のCPU利用率の初期制御値(%) p=1,2,　　, q=1,2,

【 図8 】

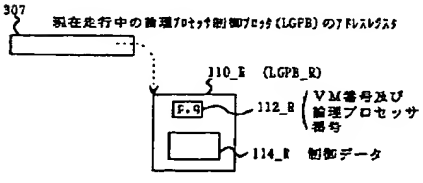
図8
論理プロセッサ制御データ





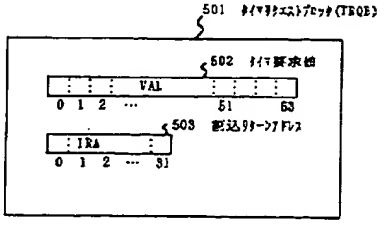
【 図10 】

図10
走行中論理プロセッサ指示レジスタ



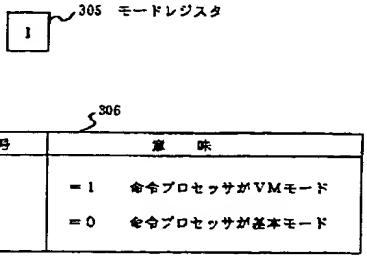
【 図11 】

図11
タイマリクエストブロック（TRQB）



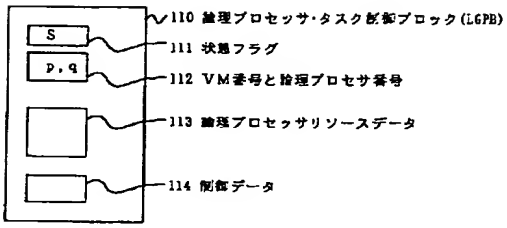
【 図19 】

図19
モードレジスタ



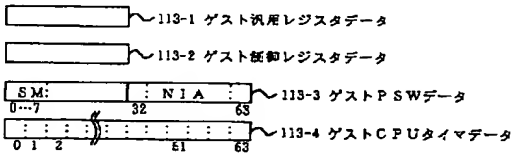
【 図7 】

図7



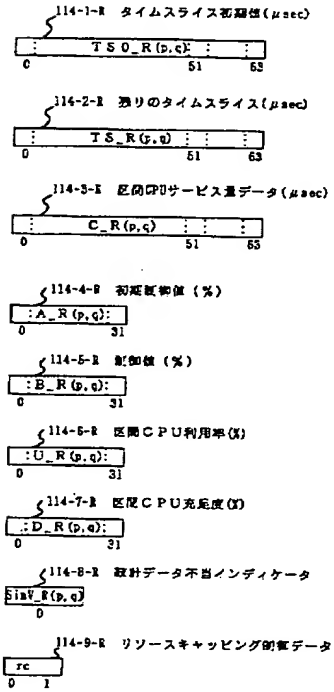
115

記　号	意　　味
S	＝0　スリープ(Sleep)状態 ＝1　レディ(Ready)状態 ＝2　アウトサービス(Out-Service)状態 ＝3　ウェイト(Wait)状態
p , q	p＝1, 2,　VM番号 q＝1, 2,　該VMにおける論理プロセッサ番号



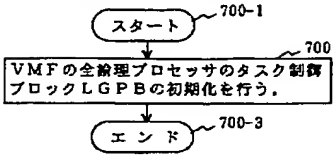
【 図9 】

図9
現在走行中論理プロセッサ制御データ



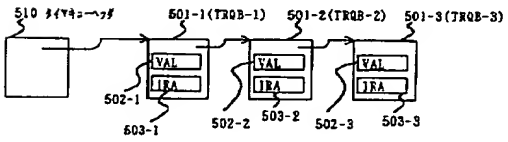
【 図24 】

図 24
ＶＭのアクティベイト



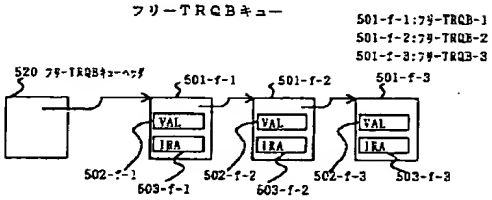
【 図12 】

図12
タイマリクエストキュー



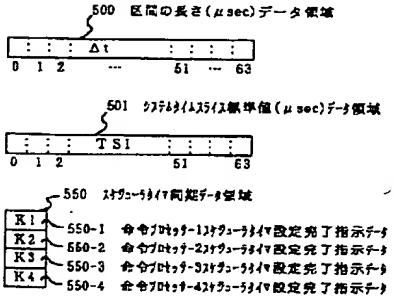
【 図13 】

図13
フリーＴＲＱＢキュー



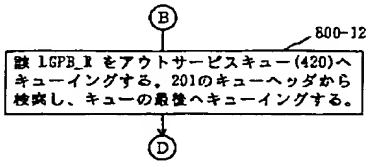
【 図14 】

図14
システム制御データ

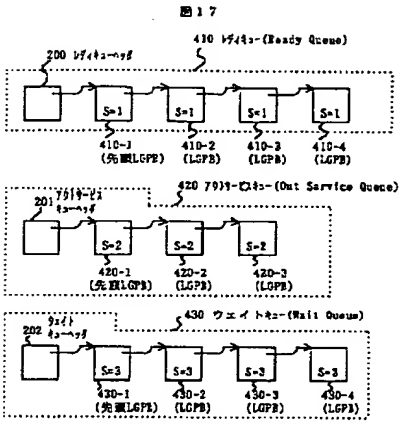


【 図29 】

図 29
論理プロセッサの中断処理 (3/3)



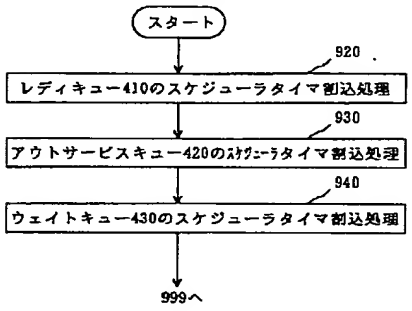
【 図17 】



【 図34 】

図 34

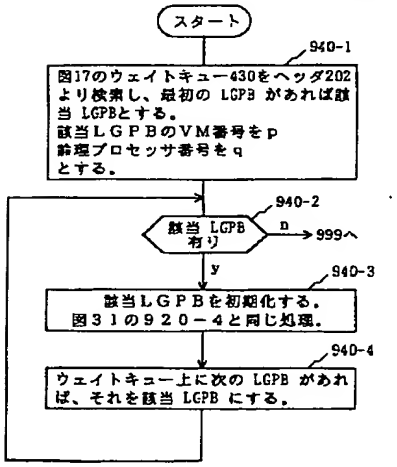
スケジューラタイマ割込処理(入口(E0)処理)



【 図37 】

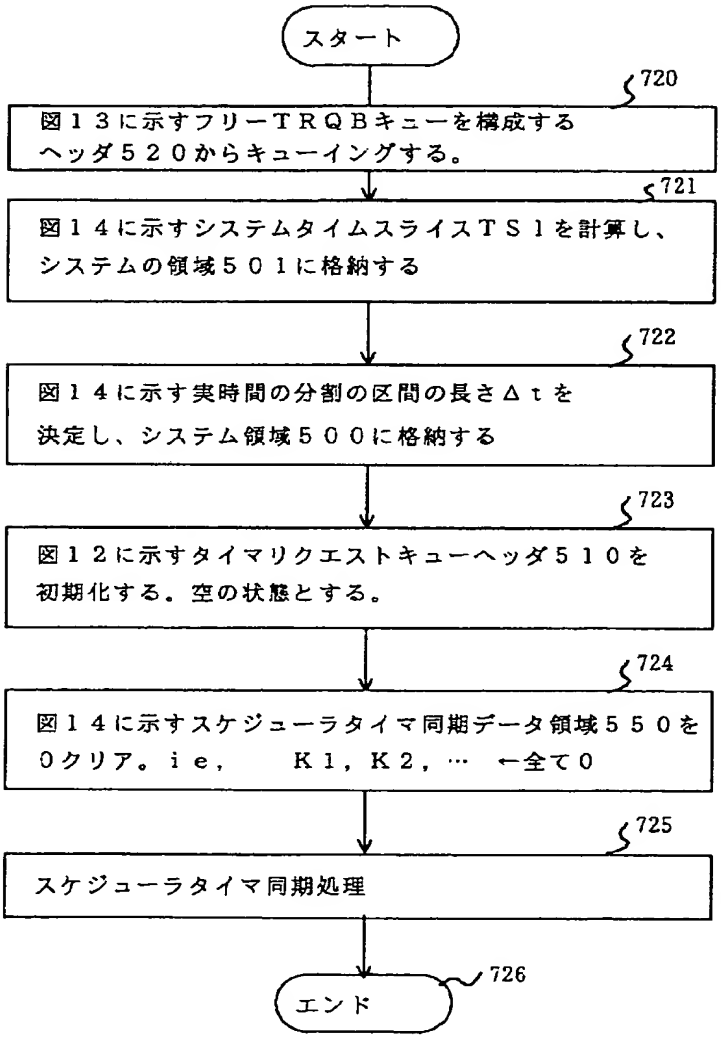
図 37

ウェイトキューのスケジューラタイマ割込処理
(940の詳細)



【 図20 】

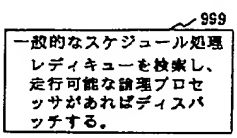
図２０
ハイパバイザ立ち上げ時の初期化



【 図38 】

図 38

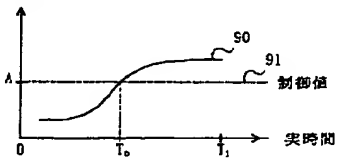
一般的なスケジューラの処理



【 図39 】

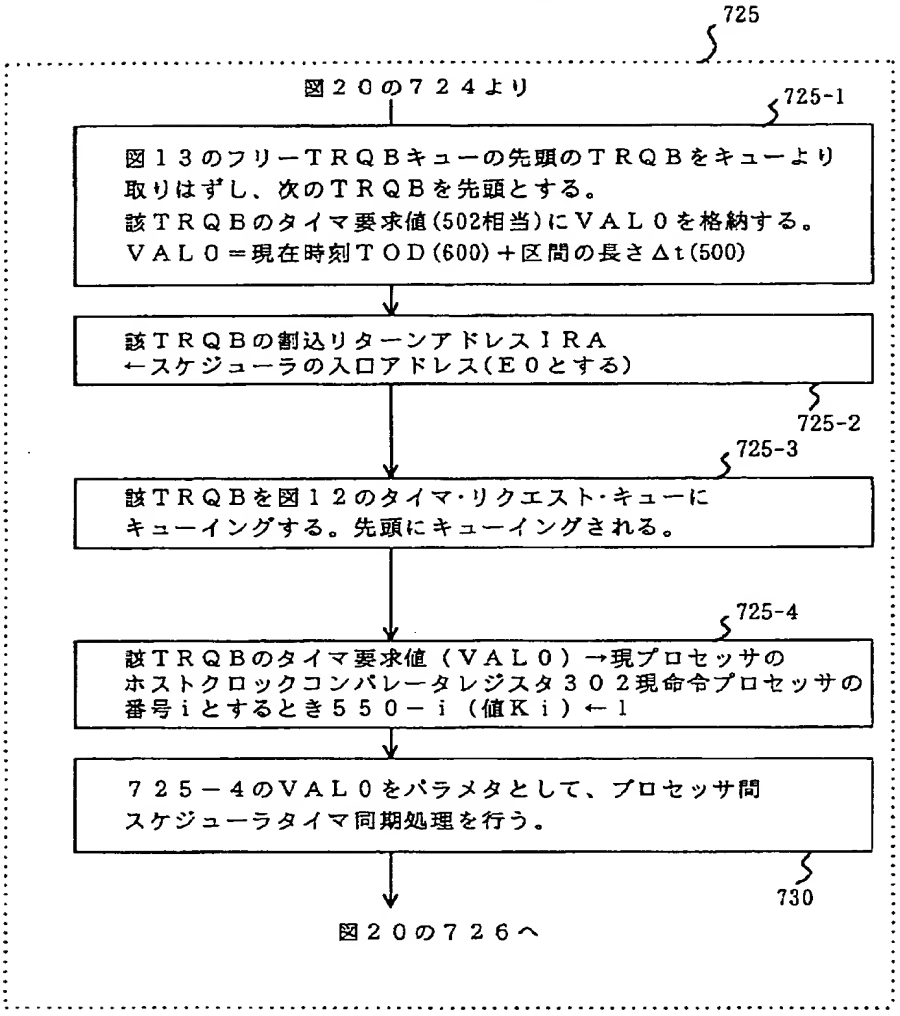
図 39

本来のＣＰＵ利用率



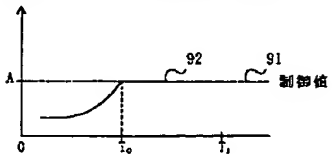
【 図21 】

図２１
スケジューラタイマ同期処理



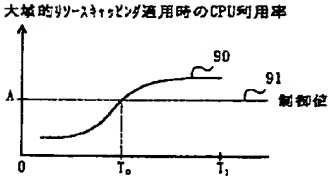
【 図40 】

図 40
局所的リソースキャッピング適用時のCPU利用率



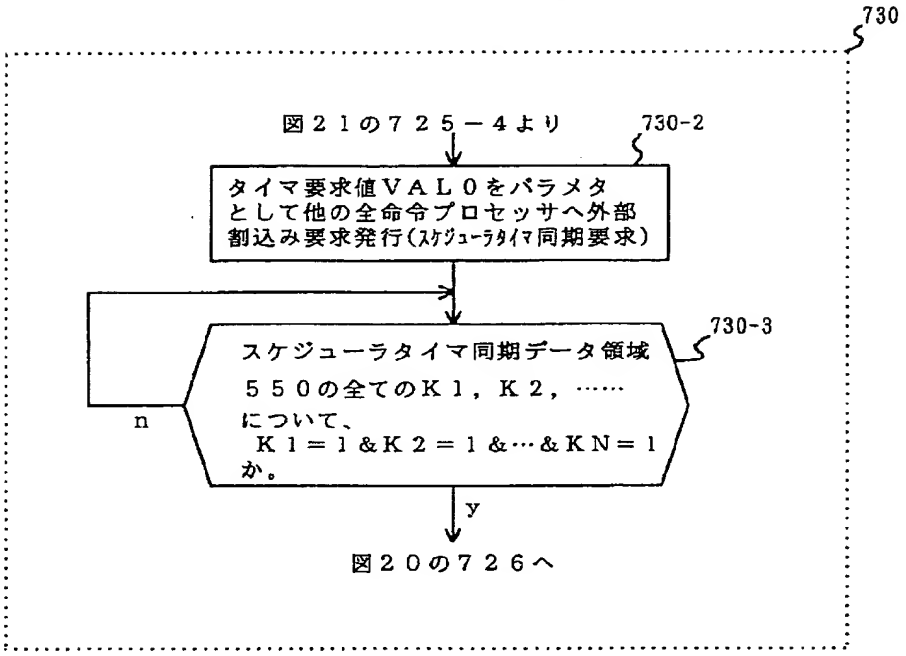
【 図41 】

図 41
大域的リソースキャッピング適用時のCPU利用率

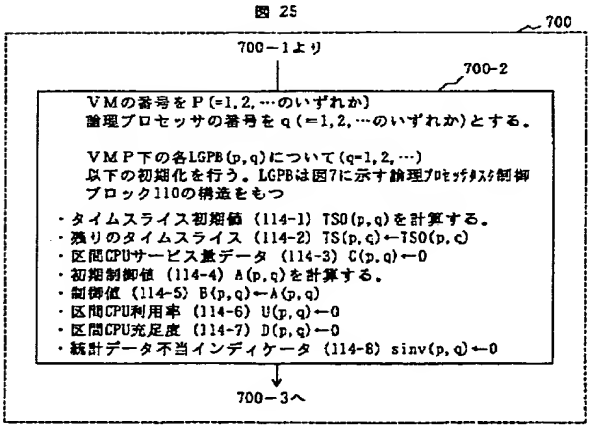


【 図2 2 】

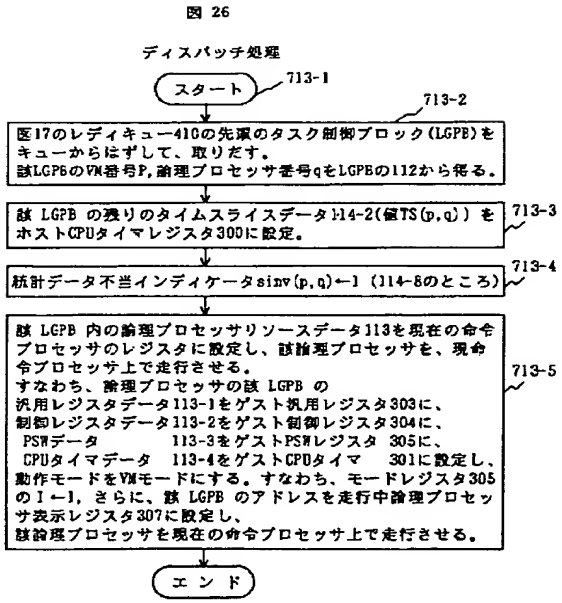
図２２
プロセッサ間スケジューラタイマ同期処理



【 図2 5 】

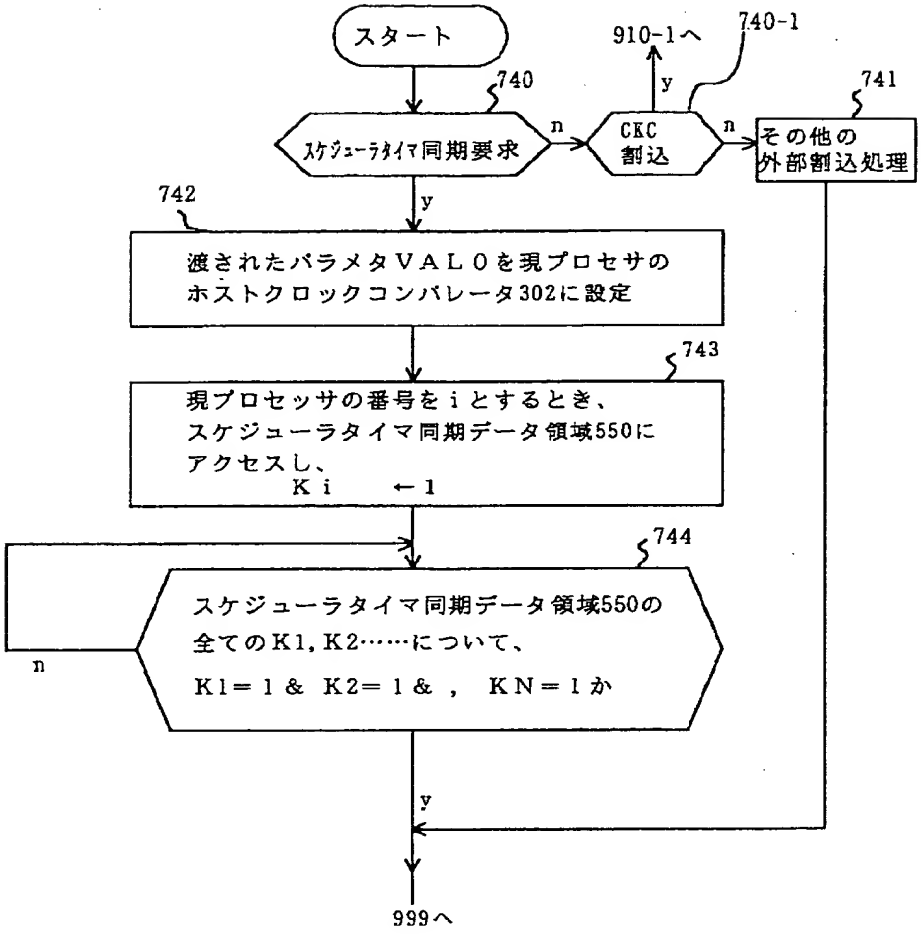


【 図2 6 】



【 図2 3 】

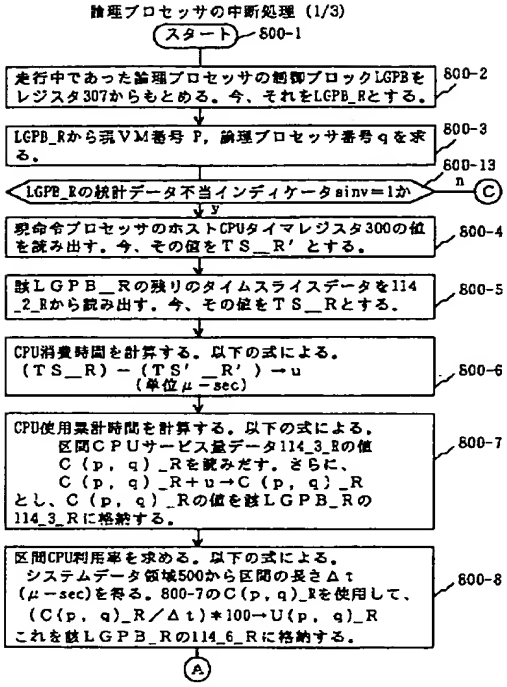
図２３
外部割込処理



【 図２７ 】

図 27

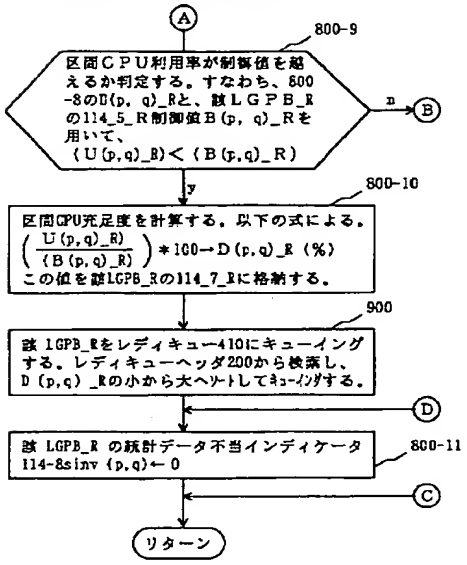
論理プロセッサの中断処理 (1/3)



【 図２８ 】

図 28

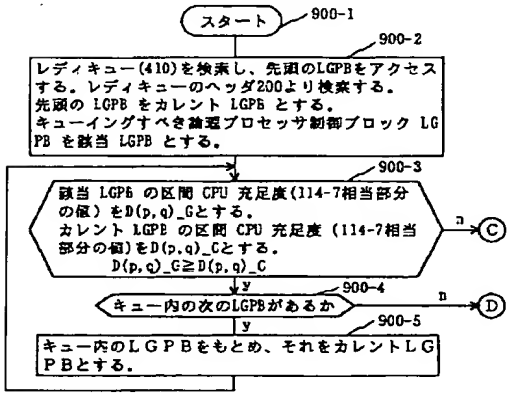
論理プロセッサの中断処理 (2/3)



【 図３０ 】

図 30

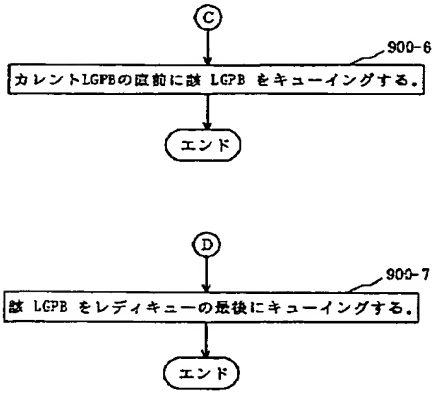
レディキューへのキューイング(900の詳細) (1/2)



【 図３１ 】

図 31

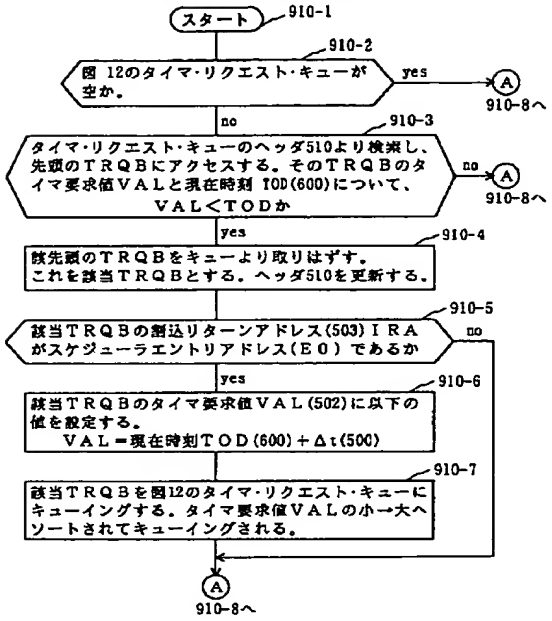
レディキューへのキューイング (2/2)



【 図３２ 】

図 32

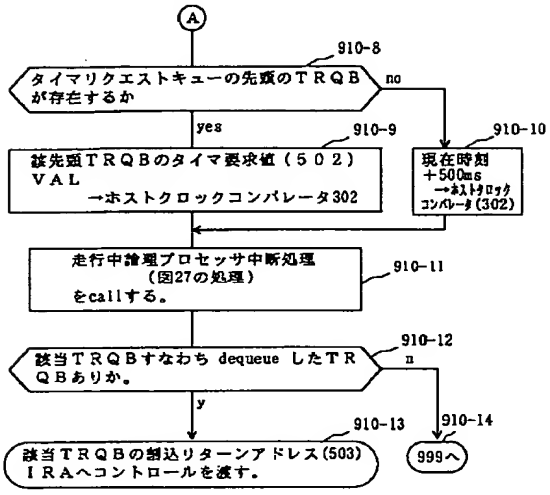
ホスト・クロック・コンパレータ割込処理 (1/2)



【 図３３ 】

図 33

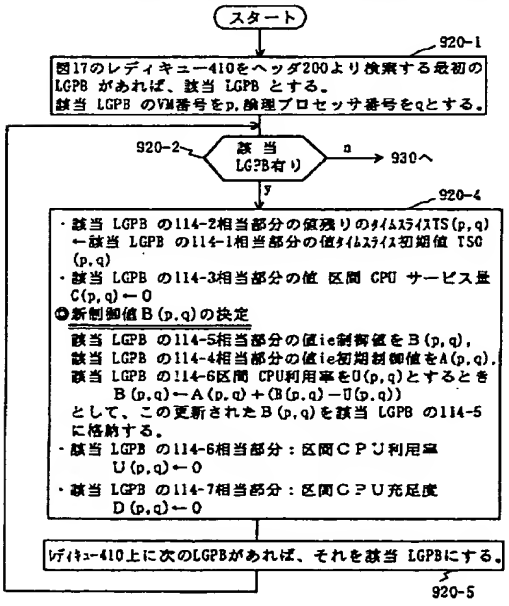
ホスト・クロック・コンパレータ割込処理 (2/2)



【 図３５ 】

図 35

レディキューに対するスケジューラタイマ割込処理(920の詳細)



【 図３６ 】

図 36

アウトサービスキューのスケジューラタイマ割込処理（930の詳細）